Springer Proceedings in Advanced Robotics 16
Series Editors: Bruno Siciliano · Oussama Khatib

Genya Ishigami
Kazuya Yoshida *Editors*

Field and Service Robotics

Results of the 12th International Conference

Springer

Springer Proceedings in Advanced Robotics 16

Series Editors

Bruno Siciliano
Dipartimento di Ingegneria Elettrica
e Tecnologie dell'Informazione
Università degli Studi di Napoli
Federico II
Napoli, Napoli, Italy

Oussama Khatib
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA, USA

Advisory Editors

Gianluca Antonelli, Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Cassino, Italy

Dieter Fox, Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA

Kensuke Harada, Engineering Science, Osaka University Engineering Science, Toyonaka, Japan

M. Ani Hsieh, GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA

Torsten Kröger, Karlsruhe Institute of Technology, Karlsruhe, Germany

Dana Kulic, University of Waterloo, Waterloo, ON, Canada

Jaeheung Park, Department of Transdisciplinary Studies, Seoul National University, Suwon, Korea (Republic of)

The Springer Proceedings in Advanced Robotics (SPAR) publishes new developments and advances in the fields of robotics research, rapidly and informally but with a high quality.

The intent is to cover all the technical contents, applications, and multidisciplinary aspects of robotics, embedded in the fields of Mechanical Engineering, Computer Science, Electrical Engineering, Mechatronics, Control, and Life Sciences, as well as the methodologies behind them.

The publications within the “Springer Proceedings in Advanced Robotics” are primarily proceedings and post-proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. Also considered for publication are edited monographs, contributed volumes and lecture notes of exceptionally high quality and interest.

An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

Indexed by WTI Frankfurt eG, zbMATH.

More information about this series at http://www.springer.com/series/15556

Genya Ishigami · Kazuya Yoshida
Editors

Field and Service Robotics

Results of the 12th International Conference

Springer

Editors
Genya Ishigami
Department of Mechanical Engineering
Keio University
Yokohama, Kanagawa, Japan

Kazuya Yoshida
Department of Aerospace Engineering
Tohoku University
Sendai, Miyagi, Japan

ISSN 2511-1256 ISSN 2511-1264 (electronic)
Springer Proceedings in Advanced Robotics
ISBN 978-981-15-9459-5 ISBN 978-981-15-9460-1 (eBook)
https://doi.org/10.1007/978-981-15-9460-1

© Springer Nature Singapore Pte Ltd. 2021
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Foreword

At the dawn of the century's third decade, robotics is reaching an elevated level of maturity and continues to benefit from the advances and innovations in its enabling technologies. These all are contributing to an unprecedented effort to bring robots to human environment in hospitals and homes; factories and schools; and in the field for fighting fires, making goods and products, picking fruits, watering the farmland, and saving time and lives. Robots today hold the promise of making a considerable impact in a wide range of real-world applications from industrial manufacturing to healthcare, transportation, and exploration of the deep space and sea. Tomorrow, robots will become pervasive and touch upon many aspects of modern life.

The Springer Tracts in Advanced Robotics (STAR) was launched in 2002 with the goal of bringing to the research community the latest advances in the robotics field based on their significance and quality. During the latest 15 years, the STAR series has featured publication of both monographs and edited collections. Among the latter, the proceedings of thematic symposia devoted to excellence in robotics research, such as ISRR, ISER, FSR, and WAFR, have been regularly included in STAR.

The expansion of our field, as well as the emergence of new research areas, has motivated us to enlarge the pool of proceedings in the STAR series in the past few years. This has ultimately led to launching a sister series in parallel to STAR. The Springer Proceedings in Advanced Robotics (SPAR) is dedicated to the timely dissemination of the latest research results presented in selected symposia and workshops.

The Twelfth edition of Field and Service Robotics edited by Genya Ishigami and Kazuya Yoshida offers a collection of a broad range of topics clustered in control, computer vision, inspection, machine learning, mapping, navigation and planning, and systems and tools. The contents of the 37 contributions represent a cross section of the current state of robotics research from one particular aspect: field and service applications and how they reflect on the theoretical basis of subsequent developments. Pursuing technologies aimed at field robots that operate in complex and dynamic environments, as well as at service robots that work closely with humans to help them with their lives, is the big challenge running throughout this focused collection.

Rich by topics and authoritative contributors, FSR culminates with this unique reference on the current developments and new directions in field and service robotics. A fine addition to the series!

Naples, Italy
Stanford, USA
December 2020

Bruno Siciliano
Oussama Khatib
SPAR Editors

Preface

FSR is a series of the biennial, single-track conferences with a specific focus on field and service applications of robotics technology. The goal of FSR is to report and encourage the development of field and service robotics. These are non-factory robots, typically mobile, that must operate in complex and dynamic environments. Typical field robotics applications include mining, agriculture, building and construction, forestry, cargo handling, and so on. Field robots may operate on the ground (of Earth or planets), under the ground, underwater, in the air, or in outer space. Service robots work closely with humans, importantly the elderly and sick, to help them with their lives. Past FSR conferences have been held successively in Australia, North America, Europe, and Japan.

This book contains the result of the 12th Conference on Field and Service Robotics (FSR 2019), held in Tokyo, Japan, from August 29th to 31st, 2019. FSR 2019 was organized by Genya Ishigami and Kazuya Yoshida as general chairs, together with the regional chairs Tim Barfoot, Marco Hutter, and Jonathan Roberts. The local arrangements and publicity of FSR 2019 were carried out by Satoko Abiko, Naohiro Uyama, Hiroki Nakanishi, and Masataku Sutoh. The organizers would like to sincerely thank the International Program Committee that generously provided their time to carry out the detailed review of all the papers: Andreas Birk, Peter Corke, Carrick Dettweiler, Philippe Giguere, Michael Jakuba, Alonzo Kelly, Jonathan Kelly, Ross Knepper, Simon Lacroix, Christian Laugier, Josh Marshall, David P Miller, Keiji Nagatani, Juan Nieto, Steve Nuske, Francois Pomerleau, Cedric Pradalier, Jonathan Roberts, Miguel Angel Salichs, Roland Siegwart, Sanjiv Singh, Gaurav Sukhatme, Salah Sukkarieh, Takashi Tsubouchi, Arto Visala, David Wettergreen, Alex Zelinsky, and Uwe Zimmer.

FSR in 2019 accepted 37 contributed papers out of 52 submitted papers, each of which was reviewed by at least two reviewers. The following five papers from those contributed papers were invited to be submitted their extended article to the Wiley Journal of Field Robotics, where the journal's regular review process underwent.

- Zhang et al., Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation
- Kolvenbach et al., Towards autonomous inspection of concrete deterioration in sewers with legged robots
- Tremblay et al., Automatic three-dimensional mapping for tree diameter measurements in inventory operations
- Kubelka et al., Radio propagation models for differential GNSS based on dense point clouds
- Dang et al., Graph-based subterranean exploration path planning using aerial and legged robots

These five papers are available in Journal of Field Robotics, Vol. 37, Issue 8.

The FSR 2019 had three excellent keynote speakers in addition to the accepted papers. We would like to acknowledge their excellent contributions to the conference:

- Davide Scaramuzza, Professors of the University of Zurich and ETH Zurich, "Are We Ready for Autonomous Drones?"
- Sanjiv Singh, Professor of Carnegie Mellon University, CEO of Near Earth Autonomy, "Field Robotics: Ready Now for Dull, Dirty and Dangerous?"
- Yoshiyuki Sankai, Professor, University of Tsukuba, President and CEO of CYBERDYNE Inc. "Challenges for Social Implementation of Innovative Cybernic Systems -Fusion of Humans, Robots and Information Systems-"

FSR 2019 would not have been possible without the generous support of our sponsors Maxell Holdings, Ltd., Nihon Binary, Kokusai Kogyo Co. Ltd., RT corporation, and Embassy of Switzerland in Japan.

Yokohama, Japan
Genya Ishigami
Kazuya Yoshida

Contents

Multi-session Lake-Shore Monitoring in Visually Challenging Conditions 1
Cédric Pradalier, Stéphanie Aravecchia, and François Pomerleau

Autonomous Visual Assistance for Robot Operations Using a Tethered UAV 15
Xuesu Xiao, Jan Dufek, and Robin R. Murphy

Tethered Dual-Wheeled Robot for Exploration of Volcanic Fumaroles on Steep Slopes 31
Keiji Nagatani and Masaki Momii

Robotic Reliability Engineering: Experience from Long-Term TritonBot Development 45
Shengye Wang, Xiao Liu, Jishen Zhao, and Henrik I. Christensen

Autonomous Driving of Six-Wheeled Dump Truck with a Retrofitted Robot 59
Tomohiro Komatsu, Yota Konno, Seiga Kiribayashi, Keiji Nagatani, Takahiro Suzuki, Kazunori Ohno, Taro Suzuki, Naoto Miyamoto, Yukinori Shibata, and Kimitaka Asano

Pre-robotic Navigation Identification of Pedestrian Crossings and Their Orientations 73
Ahmed Farid and Takafumi Matsumaru

Improvement in Measurement Area of Three-Dimensional LiDAR Using Mirrors Mounted on Mobile Robots 85
Kazuki Matsubara and Keiji Nagatani

The Hulk: Design and Development of a Weather-Proof Vehicle for Long-Term Autonomy in Outdoor Environments 101
Stephen Kyberd, Jonathan Attias, Peter Get, Paul Murcutt, Chris Prahacs, Matthew Towlson, Simon Venn, Andreia Vasconcelos, Matthew Gadd, Daniele De Martini, and Paul Newman

Effects of Turning Radius on Skid-Steered Wheeled Robot Power Consumption on Loose Soil 115
Jean-Sebastien Fiset, Meysam Effati, and Krzysztof Skonieczny

Toward Precise Robotic Grasping by Probabilistic Post-grasp Displacement Estimation 131
Jialiang (Alan) Zhao, Jacky Liang, and Oliver Kroemer

Development of HYDROVER: A Torque-Controlled Hydraulic Rover 145
Sang-Ho Hyon, Yusuke Ida, Kosuke Ueda, Junichi Ishikawa, and Minoru Hiraoka

VI-SLAM for Subterranean Environments 159
Andrew Kramer, Mike Kasper, and Christoffer Heckman

Simultaneous Localization and Mapping of Subterranean Voids with Gaussian Mixture Models 173
Wennie Tabib and Nathan Michael

Obstacle Climbing of Tracked Robot for Unfixed Cylindrical Obstacle Using Sub-tracks 189
Ryosuke Yajima and Keiji Nagatani

Multi-rover Exploration Strategies: Coverage Path Planning with Myopic Sensing 205
Mickael Laine and Kazuya Yoshida

Machine Learning Techniques for AUV Side-Scan Sonar Data Feature Extraction as Applied to Intelligent Search for Underwater Archaeological Sites 219
Nandeeka Nayak, Makoto Nara, Timmy Gambin, Zoë Wood, and Christopher M. Clark

Ridge-Tracking for Strawberry Harvesting Support Robot According to Farmer's Behavior 235
Ayanori Yorozu, Genya Ishigami, and Masaki Takahashi

ANYmal in the Field: Solving Industrial Inspection of an Offshore HVDC Platform with a Quadrupedal Robot 247
C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter

Large-Scale 3D Mapping of Subarctic Forests 261
Philippe Babin, Philippe Dandurand, Vladimír Kubelka, Philippe Giguère, and François Pomerleau

Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem 277
Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto

Fast Exploration Using Multirotors: Analysis, Planning, and Experimentation 291
Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael

Ray Tracing and Use of Shadows as Features for Determining Location in Lunar Polar Terrain 307
Eugene Fang and William “Red” Whittaker

Planning for Robotic Dry Stacking with Irregular Stones 321
Yifang Liu, Jiwon Choi, and Nils Napp

Actively Articulated Wheel-on-Limb Mobility for Traversing Europa Analogue Terrain 337
William Reid, Gareth Meirion-Griffith, Sisir Karumanchi, Blair Emanuel, Brendan Chamberlain-Simon, Joseph Bowkett, and Michael Garrett

Meander-Based River Coverage by an Autonomous Surface Vehicle 353
Nare Karapetyan, Jason Moulton, and Ioannis Rekleitis

Autonomous 3D Semantic Mapping of Coral Reefs 365
Md Modasshir, Sharmin Rahman, and Ioannis Rekleitis

Dynamic Autonomous Surface Vehicle Controls Under Changing Environmental Forces 381
Jason Moulton, Nare Karapetyan, Michail Kalaitzakis, Alberto Quattrini Li, Nikolaos Vitzilaios, and Ioannis Rekleitis

Airborne Particle Classification in LiDAR Point Clouds Using Deep Learning 395
Leo Stanislas, Julian Nubert, Daniel Dugas, Julia Nitsch, Niko Sünderhauf, Roland Siegwart, Cesar Cadena, and Thierry Peynot

Experimental Validation of Structured Receding Horizon Estimation and Control for Mobile Ground Robot Slip Compensation 411
Nathan D. Wallace, He Kong, Andrew J. Hill, and Salah Sukkarieh

Lessons Learned from Deploying Autonomous Vehicles at UC San Diego 427
David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit Binnani, and Henrik I. Christensen

Smooth Local Planning Incorporating Steering Constraints 443
Freya Fleckenstein, Wera Winterhalter, Christian Dornhege,
Cédric Pradalier, and Wolfram Burgard

A Planning Framework for Persistent, Multi-UAV Coverage with Global Deconfliction 459
Tushar Kusnur, Shohin Mukherjee, Dhruv Mauria Saxena,
Tomoya Fukami, Takayuki Koyama, Oren Salzman,
and Maxim Likhachev

About the Editors

Prof. Genya Ishigami received his Ph.D. from Tohoku University, Japan, in 2008. He was a postdoctoral associate at the Massachusetts Institute of Technology from 2008 to 2010, and a research associate at Japan Aerospace Exploration Agency from 2010 to 2013. He is currently an Associate Professor at Keio University, Japan. His research interests are in the areas of mobility analysis, vehicle-terrain interaction mechanics, perception, navigation, and control, for application to field robotics such as planetary exploration rovers, unmanned ground vehicles, and construction machines.

Prof. Kazuya Yoshida received B.E., M.S., and Dr. Eng, degrees in Mechanical Engineering Science from the Tokyo Institute of Technology, Japan, in 1984, 1986, and 1990, respectively. He served as a Research Associate of the Tokyo Institute of Technology from 1986 to 1994 and Visiting Scientist of Massachusetts Institute of Technology, USA in 1994. He is a Full-time Professor at the Department of Aerospace Engineering, Tohoku University, Japan, where he started the Space Robotics Lab. He also serves as Head of the International Joint Graduate Program for Integration of Mechanical Systems in Tohoku University since 2018. He has also been contributing to space robotics education for international students at International Space University in Strasbourg, France (for Master of Space Studies) and various locations worldwide (for Summer Study Programs).

Multi-session Lake-Shore Monitoring in Visually Challenging Conditions

Cédric Pradalier, Stéphanie Aravecchia, and François Pomerleau

1 Introduction

This paper presents the results of a large-scale experiment on multi-session monitoring of a natural environment using a 2D laser scanner onboard an autonomous surface vessel. Our dataset (Symphony Lake Dataset [8]) was recorded over 4 years by autonomously circumnavigating a small reservoir, named Lake Symphony, in Metz, France (see Fig. 1). A total of 130 surveys were realized between January 1, 2014 and November 1, 2017, corresponding to approximately 130 km of autonomous operations. During these surveys, the vessel recorded 2D laser scans, side looking images, GPS and compass. From these data, our purpose is to build a consistent map of the lake shore over time.

At a time where most mapping work is focused on 3D data (point clouds acquired from RGB-D sensors or multi-beam lidars) or visual(-inertial) SLAM, the originality of our 2D-mapping stems from the type of environment we are considering for multi-session mapping. The lake shore we are considering consists mostly of trees, reeds, shrubs and lawn, with minimal structure and no corner or straight lines. Also, because we are observing trees and shrubs from a distance of 10 m, the laser beams stochastically penetrate the environment making it semi-transparently from the laser point of view. Regarding lawn, the low ground gradient makes for grazing laser beams, which means that small variations in height or attitude of the sensor lead to large measurement variations. This is amplified by the fact that this lake is a flood

---

C. Pradalier (✉) · S. Aravecchia
Georgia Tech Lorraine-UMI 2958, GeorgiaTech-CNRS, Metz, France
e-mail: cedric.pradalier@georgiatech-metz.fr

S. Aravecchia
e-mail: stephanie.aravecchia@georgiatech-metz.fr

F. Pomerleau
Laval University, Quebec, Canada
e-mail: francois.pomerleau@ift.ulaval.ca

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_1

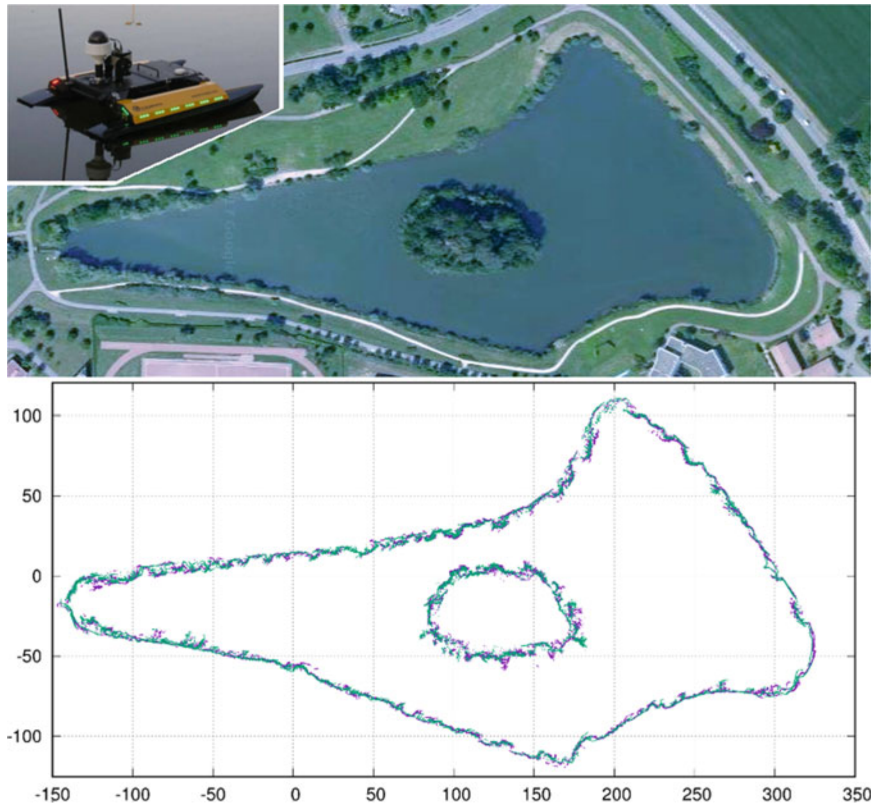


Fig. 1 Overhead imagery of the environment considered in this paper. *Insert* Kingfisher robot. *Bottom* Resulting lidar map (in meters)

buffer so the water level can vary by up to two meters in a short amount of time. These conditions, combined with the large time span covered by our dataset, make for a specific set of challenges that will be discussed in this paper.

Because we deal with a 2D laser in a natural environment, we do not enjoy the luxury of reliable feature detectors and descriptors. As a result, this work relies on matching and aligning 2D point clouds, from raw sensor measurements to local accumulated maps (keyframes) or global maps. This is the domain of Iterative Closest Point (ICP) [14]. In its principle, the ICP algorithm takes as an input a pair of point clouds and iteratively estimates the geometric transformation that projects the first point cloud to the second one. The basic assumption is that the two point clouds have a reasonable overlap, the environment is not changing too much between the two scans, and a reasonable estimate of the transformation is available to start from. The latter is particularly important if the two former assumptions cannot be guaranteed. In the right conditions, ICP can be a very fast algorithm, suitable for real time. This paper relies on the implementation available as `libpointmatcher` [15].

The contribution presented in this paper is threefold. First, we present a method for large-scale multi-survey lake-shore mapping using 2D laser, and we evaluate it on the publicly available Lake Symphony Dataset [8]. Second, the localization and 2D environment maps are added to the public dataset as a globally consistent baseline localization. Third, combining the localization result with the visual part of the dataset, we propose a 16k-image pair dataset for place recognition and change detection in a natural environment under significant seasonal changes.

2 Related Work

Our team's initial work on the Symphony Lake Dataset has been presented in [9]. So far, our objective has been to focus on the visual data to find similar views of the lake shore between different surveys and use image deformations to align them. First, a 3D model was constructed using visual-SLAM assisted with GPS for scale consistency. Second, images were aligned using first SiftFlow [12] and finally our own Reprojection Flow [9]. Initial results have been promising, but getting consistently good alignments beyond 4–6 months has proven to be a very significant challenge. It has not yet been possible to bring the full 130 surveys into alignment with these methods. A review of lake shore monitoring can be found in [9]. Terzakis et al. [17] also addresses this problem but focuses on real-time short-term visual odometry from lake-shore images.

More recently, a review of the performance of off-the-shelf visual-SLAM solutions on this dataset was conducted in [3]. This showed that dense approaches relying on the minimization of photogrammetric errors are providing the best results on the very specific images of this dataset. In particular, DSO [6] proved to be able to build good quality maps on some sub-sequences of some of the surveys, mostly in summer where the semi-transparency is not so marked. An extension of DSO has been presented in [18] to use multiple baselines and make it more robust to the conditions observed in natural environment datasets. All these works have shown that single-survey visual mapping on this dataset is a very significant challenge and that we are still far from the large-scale multi-survey mapping we are aspiring to. In comparison, this work uses a 2D lidar, which is in general a much less challenging sensor, but focuses on building the necessary robustness to work on 130 surveys, from summer to winter, flood to drought, including interaction with local wildlife, fishermen and low-level mechanical issues. For surveying purposes, images will then be compared based on the pose computed from the laser data.

Laser-based solutions for multi-session mapping were investigated before using `libpointmatcher`, but in urban environments [16]. Only three sessions during a year were used to investigate the impact of dynamic elements on the representation of the surrounding. The application of the same algorithm was demonstrated for shore mapping [10], but only with a few surveys. This paper extends on the same idea, but pushes the extent of the evaluation using two orders of magnitude of more surveys than prior work.

Finally, our approach relies on factor-graphs [4], and specific approaches for multi-session SLAM have been explored extensively (e.g., [13]). Similar to those, we build an extensive network of relations between the surveys and include them in the factor-graph optimization. However, our evaluation is the only one that focuses on a mostly natural environment dominated by vegetation.

3 Method

At a high level, our method is a three-stage process as shown in Fig. 2. Initially, every survey is processed to build an accumulated shore map as a trajectory and point cloud split into local keyframes (Sects. 3.1 and 3.2). In the second stage (Sect. 3.3), these survey maps are put in a common reference frame and at the third stage (Sects. 3.4 and 3.5), a fine-grained alignment is used to deform the individual maps into alignment. The following sections will provide details on every stage of this process as well as our evaluation method.

3.1 Intra-survey Keyframe Creation

The first step of our long-term mapping solution is the acquisition of local maps using a 2D lidar. This has been demonstrated many times in the past using lidar and ICP. In this case, ICP is applied between one laser scan and the next to estimate the 2D displacement between the laser scans. Contrary to approaches such as [2], our framework [15] implements simultaneous localization and mapping, thus accumulating lidar points into a local map. One of the challenges of boat-mounted lidar

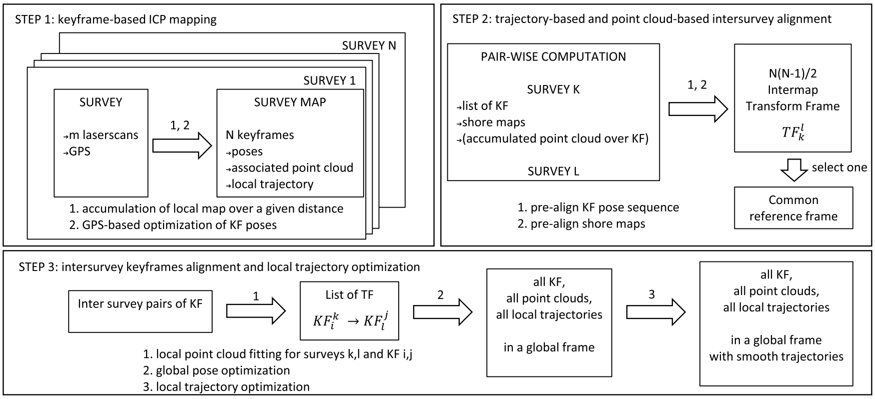

STEP 1: keyframe-based ICP mapping
SURVEY N
SURVEY 1
SURVEY
→m laserscans
→GPS
1, 2
SURVEY MAP
N keyframes
→poses
→associated point cloud
→local trajectory
1. accumulation of local map over a given distance
2. GPS-based optimization of KF poses
STEP 2: trajectory-based and point cloud-based intersurvey alignment
PAIR-WISE COMPUTATION
SURVEY K
→list of KF
→shore maps
→(accumulated point cloud over KF)
SURVEY L
1, 2
N(N-1)/2
Intermap
Transform Frame
TF_k^l
select one
Common
reference frame
1. pre-align KF pose sequence
2. pre-align shore maps
STEP 3: intersurvey keyframes alignment and local trajectory optimization
Inter survey pairs of KF
1
List of TF
$KF_i^k \rightarrow KF_l^j$
2
all KF,
all point clouds,
all local trajectories
in a global frame
3
all KF,
all point clouds,
all local trajectories
in a global frame
with smooth trajectories
1. local point cloud fitting for surveys k,l and KF i,j
2. global pose optimization
3. local trajectory optimization

Fig. 2 Overview of our method

mapping is that it is extremely hard to measure odometry (i.e., local displacement with respect to the water surface) on a slow-moving vessel. When there are enough features in the laser scans, odometry is not really needed and laser odometry works precisely. However, in our natural settings and given the large variation in conditions we are considering, laser odometry fails occasionally and another odometry source can be used to bridge the gap until the lidar sees enough of the environment to recover. We resort to the simple expedient of using GPS information as an estimate of position. At our speed (lower than 0.5 m/s), GPS-based velocity and heading estimation are rather unreliable. As a result, we complement the GPS information with heading estimated from a low-cost compass.

Regarding the mapping process, we use the dynamic mapper presented in [16]. Its main advantage, beyond being a mapping framework and not an odometry as in [2], is that it is able to estimate the dynamicity of laser points, thus focusing the alignment and localization on the most reliable and stable points in the environment. However, a lake is a specific environment because of its size with respect to the lidar perception. Lidar returns tend to come from only a small section of the shore on the side of the surveying vessel, and similar to the "infinite" corridors from [1], the trajectory and in particular the vessel heading estimate tend to drift if no globally consistent correction is applied. This correction will be presented in the next section. However, to make sure the effect of this drift does not impact the quality of the final map, we resorted to the common solution of separating the map into keyframes. These keyframes correspond to the accumulation of a local map over a number of laser scans or a given distance. These cut-off points are chosen to make sure that the orientation drift within this window is minimal, guaranteeing that the map is locally structurally consistent within keyframes. This is important from a computational perspective because all the laser scans that have been involved in the creation of keyframes can be forgotten. All their information is considered summarized in the keyframe map and the latter can be considered immune to the local trajectory adjustments in the various steps of optimization that will be presented in the following sections.

The output of this intra-survey mapping is, for each survey, a vessel trajectory, a list of keyframe coordinates and for each keyframe an associated point cloud. Combining the point clouds from all the keyframes leads to an environment map as shown in Fig. 3.

3.2 *Drift Control from GPS and Factor-Graphs*

Despite its local precision, lidar-based odometry tends to slowly drift when not provided with absolute measurements acting as anchors. This is illustrated in Fig. 3. Two types of measurements can play this role: either absolute information such as GPS or compass, or loop-closure detection. Our solution is built on a factor-graph [4] implementation using G2O [11]. The optimized variables are the poses of the keyframes ($SE(2)$). The absolute observations are the GPS measurement ($I\!R^2$), complemented with the compass observation to make it a full pose in $SE(2)$. The inter-

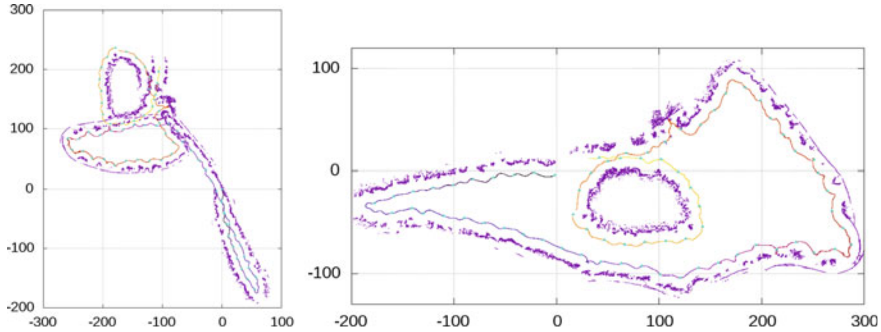


Fig. 3 ICP maps from the survey on February 2, 2016, axes in meters. *Left* map without using GPS correction, which is locally accurate but accumulates drift over the 1 km path. *Right* map with GPS correction showing global consistency because of the use of ICP and factor-graph optimization. The colored line is the trajectory, colored as a function of time

keyframe transformations from lidar odometry $SE(2)$ are used to link the estimated poses. These are estimated for the first time when a new keyframe is created by recording the position of the boat in its original keyframe at the time of the transition. The transformation is then refined when a keyframe is completed by using ICP between the keyframe point cloud and the point cloud from the previous keyframe.

An example of the resulting map is shown in Fig. 3 *Right*, while Fig. 4 gives an impression of all the maps generated at this stage.

In practice, a new keyframe starts every 20 m and this optimization is fast enough to be run in real time in the background every time a new keyframe is inserted. Because it runs continuously, most of the graph is already stabilized and G2O only requires 3–5 iterations to converge after each keyframe.

The output of this stage is the same as before but the keyframe poses have now been adjusted to account for the absolute measurement. Note that we make the assumption that the keyframes are adjusted by small increments which are not visible at the scale of the point cloud inside a keyframe. Hence, the adjustment result is not used to refine the local point cloud and the recorded trajectory inside the keyframe.

3.3 Global Map Alignment

From the previous sections, we have now constructed a set of 130 lake-shore maps (Fig. 4), all of them internally consistent up to the trade-off between the GPS and lidar-ICP precisions. It can be noted that some maps are incomplete. These are the results of the survey ending before a full circumnavigation, typically due to hardware errors. Our ultimate goal is to align all of these to a level where we can constitute pairs of images viewing the lake shore from the same pose. To this end, we intend to build a massive factor-graph optimization problem where the keyframes in all of the

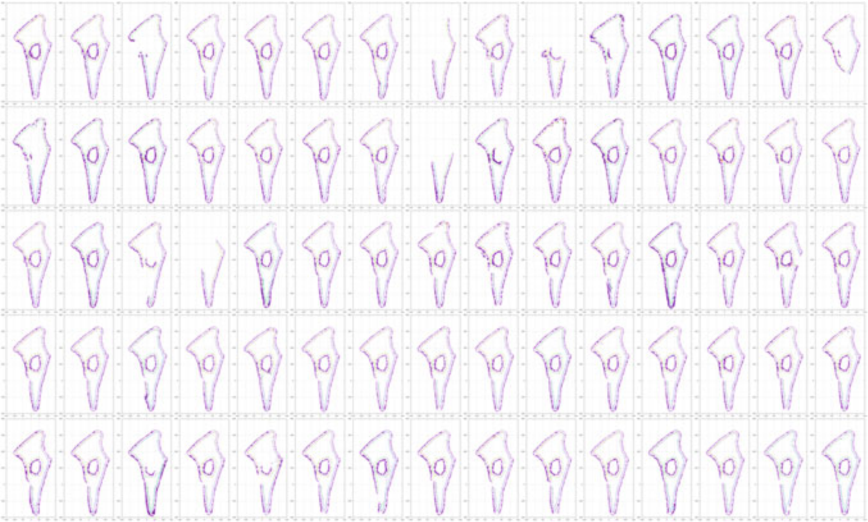


Fig. 4 All maps and trajectories after intra-survey ICP and GPS-based map stabilization

130 surveys are linked with their counterparts in other surveys with an estimate of the transform between the keyframes estimated by ICP between the keyframe point clouds. The detail of this process will be presented in Sect. 3.4. However, because the keyframe point clouds have been acquired at different times of the year, sometimes with more than 3 years of interval, and because the keyframes may have a minimal overlap, ICP does not converge reliably unless it has a reasonable initial estimate. This section presents how this estimate is obtained.

The maps obtained from intra-survey mapping (Sect. 3.2) are optimized to be close to the GPS measurements, assuming a 10 m uncertainty on GPS. Hence, to build an initial estimate of keyframe pose, we need to estimate the precise offset between the maps from two surveys. Note that, because of the GPS constraint and the size of the environment we are considering, the optimization gives very little freedom in rotation and the map offset is mostly a translation. To obtain this offset, we again use a two-stage process: first, we use ICP to align the boat trajectories considered as point clouds and starting from this offset estimate, we try to align the map point clouds. It is important to remember that the global maps are obtained by agglomerating the point clouds from all the keyframes so they may be large and noisy because of the natural environment. Running ICP directly on these maps, without the trajectory alignment, works sometimes but lacks robustness and precision. Nevertheless, the boat runs an approximate model-predictive control system to follow the lake shore at a fixed distance despite its significant dynamic. This means that trajectories tend to be very similar between surveys and very smooth. We take advantage of this by running the first alignment between the trajectories using ICP. We can then use the

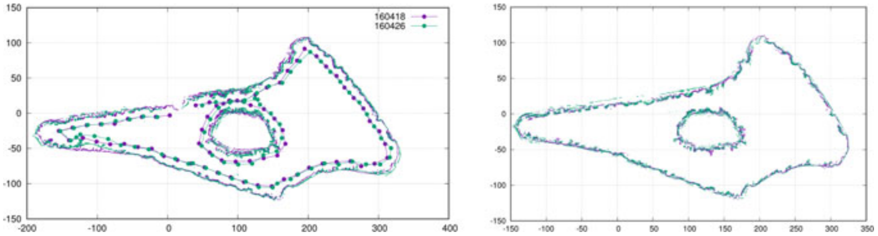


Fig. 5 *Left* Visualization of the keyframe trajectory and point clouds for two surveys (i.e., map 160418 and 160426) after global alignment. *Right* Visualization of map 140106 (in cyan) compared with 171030 (in purple) after inter-survey map alignment. The lines on the top part of 140106 are shorelines visible because of the high water level on this day

resulting transformation between trajectories as an initial estimate for running ICP between the global point clouds.

After this level of alignment of the surveys, the map point clouds are globally aligned but they may have a section where the map orientation drifted and was not corrected by the GPS measurements. Figure 5 shows the aligned keyframe trajectory and point clouds, despite the global alignment local drift still being visible.

At the end of this stage, we have obtained all our keyframes in a common reference frame where they can be compared. The trajectory and the point clouds have not been modified.

3.4 Iterative Multi-session SLAM

At this stage, all the keyframes from all the surveys are now in a common reference frame. In order to align all the surveys and build a common map combining information from all the surveys, we will resort once again to a large factor-graph structure. The variables to be optimized will be the keyframe poses in the common frame. The intra-survey information will provide an estimate of the transform between successive keyframes within a survey, as well as GPS estimates. What remain to be estimated are transforms between pairs of keyframes in different surveys. Non-consecutive pairs from a given survey are also considered here. They correspond to situations where the boat revisits the same place at two different times in the survey.

To estimate the transform between a pair of keyframes, we will take advantage of our ICP framework and try to align the keyframe point clouds. To achieve satisfactory robustness, we need to be very strict and require at least 70% of overlap between the aligned point clouds. In our setup, we have 130 surveys, each containing the order of 80 keyframes, resulting in approximately 10^4 poses to optimize, and naively 10^8 ICP pairs to consider.

In order to make the estimation of ICP links tractable, we take advantage of the common reference frame and for each keyframe we use the nearest neighbor search to find keyframes on which point clouds we can apply ICP. Note that all these ICP instances are run using the estimated keyframe pose in the common reference frame as an initial guess of the transformation.

Once all the keyframe links have been added to the factor-graph problem, the optimization can be driven to its minimum. This optimization locally deforms the map from every survey to match all the constraints from intra- and inter-survey ICP matches while accounting for the GPS measurements. The tuning of the relative weights is tricky here but we consider that the intra-survey transformation from consecutive frames are the most reliable; inter-survey transformations are a bit less reliable but still much better than GPS.

However, because the survey maps are locally modified, these deformations can lead to a new set of neighbors to every keyframe. Among these new neighbors may be some for which the transformation can be estimated by ICP. To account for this new information, the neighbor search and optimization are run iteratively: using the new pose estimates, a new KD-tree [5] is built, out of which new set of neighbors are extracted. ICP is then applied to those that have not been considered yet and a new factor-graph problem is built and optimized. The iterations stop when no new neighbors can be found. In our setup, every iteration adds approximately 10% of the number of neighbors found by the previous iteration, leading to a bit more than 10^7 ICP evaluation among which 1.4×10^6 are found to have more than 70% of overlap and are used in the final factor-graph problem.

Figure 5 shows an example of pairs of aligned maps reconstructed from the optimized keyframes. Figure 8 shows the consistency of the reconstruction in a map containing all the point clouds from the 130 surveys.

The output of this stage is the optimized poses of all the keyframes in a common reference frame, as well as the point cloud map resulting from every survey.

3.5 *Local Trajectory Optimization*

As mentioned earlier, the optimization stages from the previous sections only affect the keyframe poses but do not adjust the boat trajectory within the keyframes. In this section, we detail how the trajectory can be adjusted to make sure it smoothly goes from the origin of a keyframe to the origin of the next one.

The ICP-based odometry from Sect. 3.1 provides a set of poses originating at the origin of a keyframe and reaching the next keyframe. These are the variables we intend to optimize in this section. In addition to making sure that the end of the local trajectory reaches the next keyframe, we also take the opportunity of smoothing the trajectory during this stage. The lidar poses are acquired at 40 Hz on a vessel moving at most at 0.3 m/s, so there is less than 1 cm of displacement between lidar scans. The boat has very little control authority in the rotation due to the water resistance and propeller arrangement so the rotation between lidar scans is rarely larger than 0.01 rad. Despite these mechanical constraints, lidar measurement noise

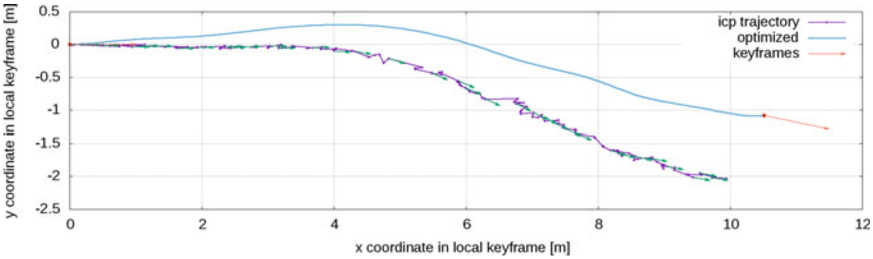


Fig. 6 Example of local trajectory optimization between two keyframes identified by the red dot and arrows. The purple sequence of the pose is characteristic of unfiltered ICP result while mapping

and, in particular, the semi-transparency of the environment, make the ICP somewhat noisy at the 5–10 cm scale.

To smooth the ICP trajectory, we took inspiration from [7] and decided to first approximate the local trajectory by a cubic B-spline with regularly sampled control points every 1 s (i.e., consistent with the low dynamic of our system). The B-spline approximation can be conducted on every $SE(2)$ pose parameter independently and then a linear correction can be added to make sure the end pose reaches the next keyframe. Figure 6 illustrates this process. Note that because these are the trajectories of a boat instead of a wheeled vehicle, the boat heading is not necessarily exactly aligned with the local tangent to the trajectory. At the end of this stage, we now have all the keyframes, point cloud maps and boat poses in a globally consistent frame with smooth trajectories.

3.6 *Image Pair Selection*

One use of the trajectories from the multi-session localization and mapping framework described above is to select images looking at the same shore point at different times. From a computer vision point of view, it also helps if the images are taken from the same viewpoint or at least from the same optical axis in order to reduce the effect of parallax. Given two optimized and smooth trajectories, it is straightforward to select a pair of nearest positions. Intersecting the optical axis of a camera with the trajectory in the second survey gives us neighboring poses with near-identical optical axes. It is then possible to use a homography to virtually rotate the two viewpoints so as to make the image planes parallel.

An example of the resulting images can be seen in Fig. 7. The perspective transformation of the selected image is made visible by the black pixels which were outside of the camera field of view in the untransformed images.

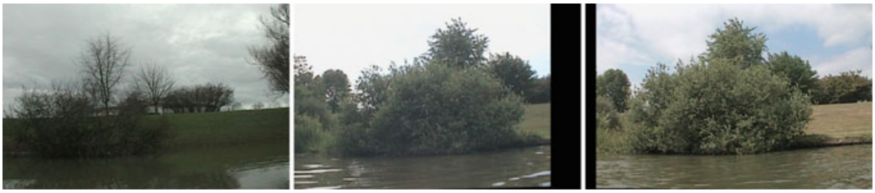


Fig. 7 Matching images from different days. *Left* image of reference taken on January 6, 2014. *Middle* and *Right* matched and rotated images taken, respectively, on June 13 and June 25, 2014

4 Results and Experiments

4.1 Global Map Alignment

The main challenge of multi-session mapping over a large time span is to maintain a consistent map. We illustrate the performance of our approach by superimposing the point clouds of a map pair in Fig. 5 and of all the 130 maps in Fig. 8. As can be observed, the local details are clearly represented and aligned precisely in the pair-wise figure. With that many maps, the level of noise increases significantly. A lot of dynamic features in open water can be attributed to wildlife interacting with the boat. On the shore, the noise level is due to uncertainty in the ICP output combined with a change of appearance resulting from variation in the water level.

Our ICP library (`libpointmatcher`) also estimates if a point is dynamic or static based on how often a laser beam goes "through" it (see [16] for details). In our particular environment, most points belong to reeds, bushes or shrubs which are semi-transparent from the laser point of view. As a result of this complexity (often resulting in "ghost" points), most points are considered dynamic using the method cited above. When plotting the histogram of the log-likelihood of a point being dynamic, we observe a distribution between -4 and -14. Selecting only the point with a dynamic likelihood below 10^{-9} results in a qualitative map improvement depicted in Fig. 8. The thinness of the shorelines across seasons highlights the precision of the performance of the iterative global alignment proposed in this paper.

4.2 Image Pairs Dataset

As mentioned earlier, the Symphony Lake Dataset was not acquired with a high precision localization system, so there is no direct ground truth to compare our multi-session SLAM results to. In order to provide a quantitative metric anyway, we elected to work on the lake-shore images. To this end, we built a dataset of image pairs by selecting randomly a pair of surveys, a reference view in the first survey and then using the approach presented in Sect. 3.6 to find the corresponding rotation-

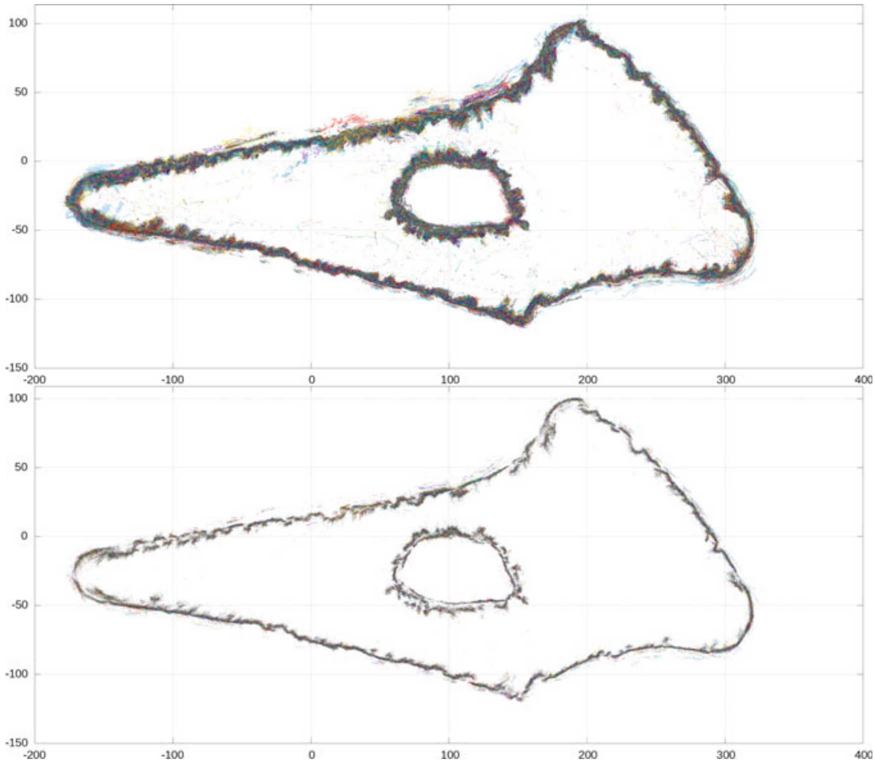


Fig. 8 Final survey combining all maps. *Top* unfiltered point clouds. *Bottom* filtered maps keeping only points with a low likelihood of dynamicity. The consistency and level of detail is particularly visible on the island where the visible shoreline is mostly vegetation

compensated image in the second survey. The two resulting images are then presented to a user who needs to click on a point in each image representing the same physical location in the world. Because the work presented here is only concerned with 2D, we can get an estimate of the alignment error by comparing the horizontal coordinates of the clicked points. With a perfectly consistent pose across surveys, and neglecting translation inaccuracies, the horizontal coordinates of physical objects should not change after compensating for the boat rotation with the homography. Note that the pixel coordinate error can be converted to angular error using the camera intrinsics. For reference, our camera has 704 pixels to cover a 42° field of view.

The histogram of the distribution of the alignment error is depicted in Fig. 9. This was computed by hand-labeling 265 image pairs, out of which four were considered too uncertain to be included. These are images with very heavy seasonal changes, lighting changes or sun glares, for which the authors were not able to identify precisely a matching physical place. This does not necessarily mean that the alignment is incorrect so these four images were removed from the statistics. Out of the remaining 261 images, the median error is 0.00°, the mean -0.25° with a standard deviation

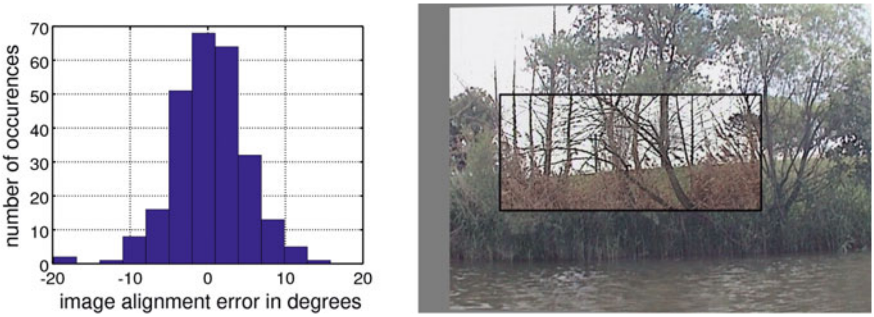


Fig. 9 *Left* Image alignment error distribution in degree for 265 random image pairs across multiple seasons. *Right* Example of a perfect alignment with an automatically generated winter insert on a rotation-compensated summer image

of 4.7°. Overall, 90% of the pairs exhibit an absolute alignment error lower than 8°. The composite picture in Fig. 9 shows how high quality of localization can be used to highlight seasonal changes in this natural environment dataset.

Because of the significant challenges observed when hand-labeling these image pairs, a dataset of 16k image pairs generated using the above approach has been added to the Symphony Lake Dataset [8] web page.[1] This dataset covers a very wide range of appearance change of this natural environment over a span of 4 years and it could be particularly useful to evaluate place-recognition algorithms. Additionally, after considering the image alignment precision evaluated in this section, the maps and boat trajectories have also been added to the Symphony Lake Dataset web page and can be used as a reference localization system by others interested in working with this challenging dataset.

5 Conclusion

In this paper, we presented our approach to large-scale multi-session SLAM using a 2D laser and ICP in a natural environment. The end-to-end approach uses ICP at multiple scales from local keyframe construction from individual laser scans to inter-survey keyframe alignment. The end results produce a set of globally consistent 2D maps and trajectories which are precise enough to select rotation-compensated images of the lake shore over multiple seasons. These results have been used to add reference maps and localization to the publicly available Lake Symphony Dataset as well as to create a new dataset of pairs of images of the lake shore that may be used for the evaluation and training of natural place-recognition algorithms. From the stepping stone presented in this paper, we intend to keep exploring Visual-SLAM algorithms

---

[1] https://dream.georgiatech-metz.fr/research-projects/symphony-lake-dataset-image-pair-dataset/.

that can work with our images. Automated image alignment (without laser), change detection and change segmentation are also topics that may be investigated with the help of this visually challenging environment monitoring dataset.

References

1. Bosse, M.C.: ATLAS: a framework for large scale automated mapping and localization. Ph.D. thesis, Massachusetts Institute of Technology (2004)
2. Censi, A.: An ICP variant using a point-to-line metric. In: IEEE International Conference on Robotics and Automation, 2008. ICRA 2008, pp. 19–25. IEEE (2008)
3. Chahine, G., Pradalier, C.: Survey of monocular SLAM algorithms in natural environments. In: Conference on Computer and Robot Vision (2018)
4. Dellaert, F., Kaess, M., et al.: Factor graphs for robot perception. Found. Trends® Robot. **6**(1–2), 1–139 (2017)
5. Elseberg, J., Magnenat, S., Siegwart, R., Nüchter, A.: Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. J. Softw. Eng. Robot. (JOSER) **3**(1), 2–12 (2012)
6. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Trans. Pattern Anal. Mach. Intell. **4** (2017)
7. Furgale, P., Tong, C.H., Barfoot, T.D., Sibley, G.: Continuous-time batch trajectory estimation using temporal basis functions. Int. J. Robot. Res. **34**(14), 1688–1710 (2015)
8. Griffith, S., Chahine, G., Pradalier, C.: Symphony lake dataset. Int. J. Robot. Res. **36**, 1151–1158 (2017)
9. Griffith, S., Pradalier, C.: Survey registration for long-term lakeshore monitoring. J. Field Robot. (2016)
10. Hitz, G., Pomerlesau, F., Colas, F., Siegwart, R.: State estimation for shore monitoring using an autonomous surface vessel. In: Experimental Robotics, pp. 745–760. Springer (2016)
11. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2O: A general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 3607–3613. IEEE (2011)
12. Liu, C., Yuen, J., Torralba, A.: SIFT flow: dense correspondence across scenes and its applications. PAMI **33**(5), 978–994 (2011)
13. Mühlfellner, P., Bürki, M., Bosse, M., Derendarz, W., Philippsen, R., Furgale, P.: Summary maps for lifelong visual localization. J. Field Robot. (2015)
14. Pomerleau, F.: Applied registration for robotics: Methodology and tools for ICP-like algorithms. Ph.D. thesis, ETH Zurich (2013)
15. Pomerleau, F., Colas, F., Siegwart, R., Magnenat, S.: Comparing ICP variants on real-world data sets. Auton Robots **34**(3), 133–148 (2013)
16. Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R.: Long-term 3D map maintenance in dynamic environments. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3712–3719. IEEE (2014)
17. Terzakis, G., Polvara, R., Sharma, S., Culverhouse, P., Sutton, R.: Monocular visual odometry for an unmanned sea-surface vehicle. arXiv preprint arXiv:1707.04444 (2017)
18. Wu, X., Pradalier, C.: Multi-scale direct sparse visual odometry for large-scale natural environment. In: International Conference on 3D Vision (2018)

Autonomous Visual Assistance for Robot Operations Using a Tethered UAV

Xuesu Xiao, Jan Dufek, and Robin R. Murphy

1 Introduction

Tele-operated robots are still widely used in Dangerous, Dirty, and Dull (DDD) environments where human presence is extremely difficult or impossible, due to those environments' mission-critical task execution and current technological limitations. Projecting human presence in remote environments is still an effective approach to leverage current technologies and actual field demand. However, one major challenge of tele-operation is the insufficient situational awareness of the remote field, caused by the onboard sensing limitations, such as relatively stationary and limited field of view and lack of depth perception from the robot's onboard camera. Therefore, the emerging state of the practice for nuclear operations, bomb squad, disaster robots, and other domains with novel tasks or highly occluded environments is to use two robots, a primary and a secondary that acts as a visual assistant to overcome the perceptual limitations of the sensors by providing an external viewpoint.

However, the usage of tele-operated visual assistants also causes problems: it requires an extra human operator, or even an operating crew, to tele-operate the secondary visual assistant. Human operators also tend to choose suboptimal viewpoints based on experience only. Most importantly, communication between the two operators of the primary and secondary robots requires extra teamwork demand, in

---

X. Xiao (✉)
Department of Computer Science, University of Texas at Austin,
Austin, USA
e-mail: xiao@cs.utexas.ed

J. Dufek · R. R. Murphy
Department of Computer Science and Engineering, Texas A&M University,
College Station, USA
e-mail: dufek@tamu.edu

R. R. Murphy
e-mail: robin.r.murphy@tamu.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_2

addition to the task and perceptual demands of the tele-operation. In this research, an autonomous tethered aerial visual assistant is developed to replace the secondary robot and its operator, reducing the human-robot ratio from 2:2 to 1:2. The co-robots team will then consist of one tele-operated primary ground robot, one autonomous aerial visual assistant, and one human operator, whose situational awareness is maintained by the visual feedback streamed from a series of optimal viewpoints for the particular tele-operation task.

Unmanned Ground Vehicles (UGVs) are stable, reliable, durable, and can thus represent humans to actuate upon the real world, while Unmanned Aerial Vehicles (UAVs) have superior mobility and workspace coverage and therefore are capable of providing enhanced situational awareness [7]. Researchers have looked into utilizing the advantages and avoiding the disadvantages by teaming up the two types of robots [1, 3]. A more relevant area was to use a UAV to augment the UGV's perception or assist UGV's task execution, such as "an eye in the sky" for UGV localization [2], providing stationary third-person view for construction machines [6], improving navigation in case of GPS loss [5], and UGV control with UAV's visual feedback [10, 16] using differential flatness [9]. However, instead of prior works' flight path execution in wide-open space or hovering at a stationary and elevated viewpoint, our aerial visual assistant needs to navigate through unstructured or confined spaces in order to provide visual assistance to the UGV operator from a series of good viewpoints. It is able to reason about the motion execution risk in complex environments and plan a path that provides good visual assistance. In particular, this work uses a tethered UAV, with the purpose of matching its battery duration with UGV's and as a fail-safe in case of malfunction in mission-critical tasks. Our tethered visual assistant utilizes the advantages and mitigates the disadvantages of the tether, in terms of tether-based indoor localization, motion primitives, and environment contact planning.

The remainder of this article is organized as follows: Sect. 2 presents the heterogeneous co-robots team. The high-level visual assistance components are described in Sect. 3, while low-level tether-based motion implementation in Sect. 4. System demonstrations are provided in Sect. 5. Section 6 concludes the paper.

2 Co-Robots Team

This section presents the co-robots team: a tele-operated ground primary robot, an autonomous tethered aerial visual assistant, and a human operator of the primary robot under the visual assistance of the aerial vehicle (Fig. 1).

2.1 Tele-Operated Ground Primary Robot

In the co-robots team, the primary robot is a tele-operated Endeavor PackBot 510 (Fig. 1 upper left). PackBot has a chassis with two main differential treads that allow

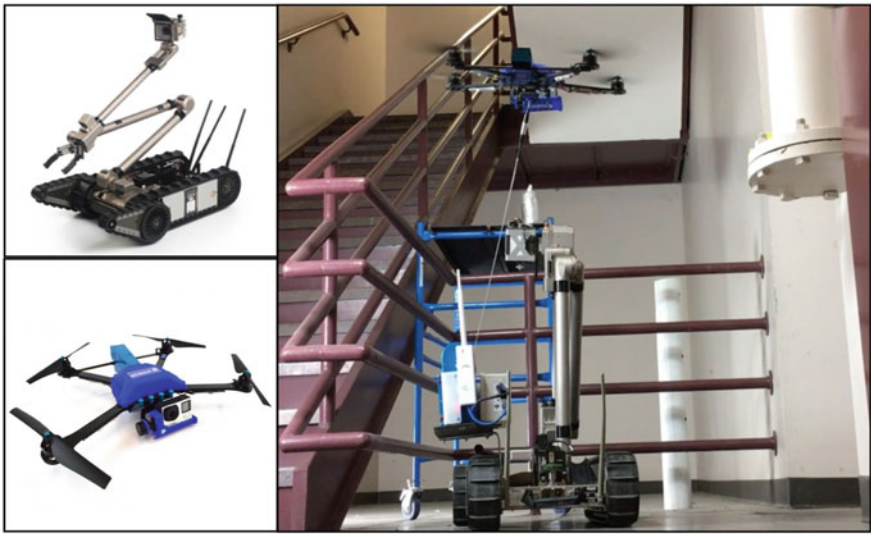
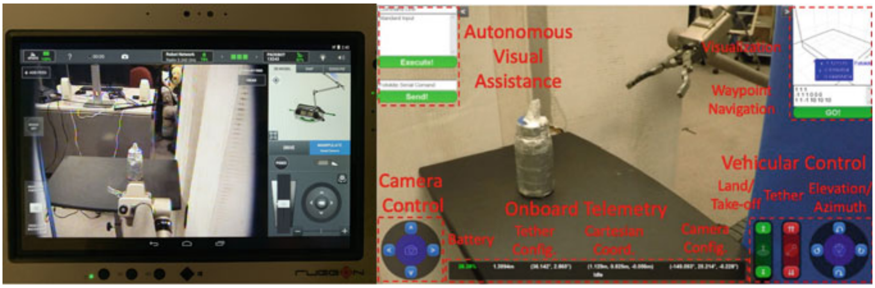


Fig. 1 The Co-Robots Team: tele-operated primary robot, Endeavor PackBot 510 (upper left), and autonomous tethered visual assistant, Fotokite Pro (lower left), picking up a sensor and dropping it into a radiation pipe in a confined staircase (right)

zero radius turn and maximum speed up to 9.3 km/h. Two articulated flippers with treads are used to climb over obstacles or stairs (up to 40°). PackBot's three-link manipulator locates on the topic of the chassis, with an articulated gripper on the second link and an onboard camera on the third. The manipulator can lift 5 kg at full extension and 20 kg close in. Motor encoders on the arm provide a precise position of the articulated joints, including the gripper, the default visual assistance point of interest. Four onboard cameras provide first-person-views, but are all limited to the robot body. On the chassis, a Velodyne Puck LiDAR constantly scans the 3-D environments, providing the map for the co-robots team to navigate through. The map does not necessarily need to be global, with the unknown parts being assumed as obstacles. Four BB-2590 batteries provide up to 8 hrs run time.

2.2 Autonomous Aerial Visual Assistant

A tethered UAV, Fotokite Pro, is used as the autonomous aerial visual assistant (Fig. 1 lower left). It could be deployed from a landing platform mounted on the ground robot's chassis. The UAV is equipped with an onboard camera with a 2-DoF gimbal (pitch and roll). The camera's yaw is controlled dependently by the vehicular yaw. The main purpose of the tether is to match the run time of the aerial visual assistant with that of the ground primary robot, since flight power could be transmitted via the


Execute!
Send!
Autonomous Visual Assistance
Visualization
Waypoint Navigation
GO!
Vehicular Control
Camera Control
Onboard Telemetry
Land/ Take-off
Tether
Elevation/ Azimuth
Battery
Tether Config.
Cartesian Coord.
Camera Config.

(a) PackBot uPoint Controller Interface (b) Visual Assistant Interface

Fig. 2 Interfaces with the human operator

tether. Additionally, the tether serves as a fail-safe in mission-critical environments. The UAV's flight controller is based on the tether sensory feedback, including the tether length, azimuth, and elevation angles. The six-dimensional coverage of the workspace makes the UAV suitable for the visual assistance purpose.

2.3 Human Operator

The human operator tele-operates the primary ground robot with the visual assistance of the UAV. In addition to the default PackBot uPoint controller with onboard first-person-view, the visual feedback from the visual assistant's onboard camera is also available for enhanced situational awareness. For example, the visual assistant could move to a location perpendicular to the tele-operation action, providing extra depth perception to the operator. The visual assistant could be either manually controlled or automated. For the focus of this research, autonomous visual assistance, a 3-D map is provided by the primary robot's LiDAR, and a risk-aware path is planned using a pre-established viewpoint quality map (discussed in the following sections). The uPoint tele-operation and visual assistance interfaces are shown in Fig. 2.

3 Visual Assistance Components

This section introduces the key components of autonomous visual assistance, including a viewpoint quality map based on the cognitive science concept of affordances, an explicit path risk representation with a focus on unstructured or confined environments, and a planning framework to balance the trade-off between reward (viewpoint quality) and risk (motion execution).

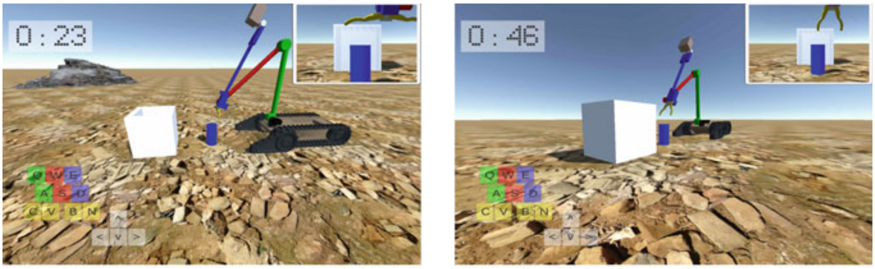
3.1 Viewpoint Quality Reward

The cognitive science concept of affordances is used to determine viewpoint quality, where the potential for an action can be directly perceived without knowing intent or models, and thus is universal to all robots and tasks [8]. For this work, four affordances are used: manipulability, passability, reachability, and traversability (Fig. 3). In order to determine the viewpoint quality (reward) for each affordance, we use a computer simulation to collect performance data with 30 professional PackBot operators. A hemisphere centered at each affordance is created, with 30 viewpoints evenly distributed on it. The 30 viewpoints are divided into five groups: left, right, front, back, and above the affordance. For each affordance, every test subject is randomly given one viewpoint within each of the five groups and tries to finish the tele-operation task in an error-free and fast manner. The number of errors, such as colliding with the wall for passability or falling off the ledge for traversability, and the completion time are recorded. The average value of the product of error and time collected by all subjects is the metric to reflect the viewpoint quality. Given any point in the 3-D space, its viewpoint reward is assigned to be the viewpoint quality of the closest point on the hemisphere. This study is still ongoing and its results will be reported in a separate paper.

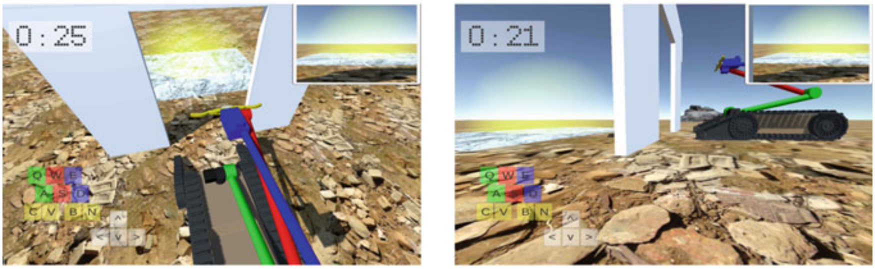
3.2 Explicit Risk Representation

In contrast to the traditional state-dependent risk representation or probabilistic uncertainty modeling, this work uses an explicit risk representation as a function of the entire path. The workspace $\mathbb{W}$ of the robot could be constructed by an occupancy grid from the LiDAR, excluding a set of obstacles $\mathbb{OB} = \{ob_i | i = 1, 2, ..., o\}$, where o is the number of obstacles. Given a start location of the visual assistant S, a simple path P could be defined as $P = \{s_0, s_1, ..., s_n\}$ where s_i denotes the ith state on the path P while $s_0 = S, \forall 1 \leq i \leq n, 1 \leq j \leq o, s_i \cap ob_j = \emptyset$, and $\forall i \neq j, 1 \leq i, j \leq n, s_i \neq s_j$. In a conventional state-dependent risk representation, risk at state s_i is defined based on a function mapping from one state to a risk index $r : s_i \mapsto ri$ and the risk of executing a path P is a simple summation of all individual states $risk(P) = \sum_{i=1}^{n} r(s_i)$. In the proposed path-dependent risk representation, however, risk at state s_i can be evaluated by not only the state, but also the path leading to s_i, $P_i = \{s_0, s_1, ..., s_i\}$ and the risk at s_i is computed through the mapping $R : (s_0, s_2, ..., s_i) \mapsto ri$. The path-level risk is relaxed from the simple summation to a more general representation $risk(P) = risk(s_0, s_1, ..., s_i)$.

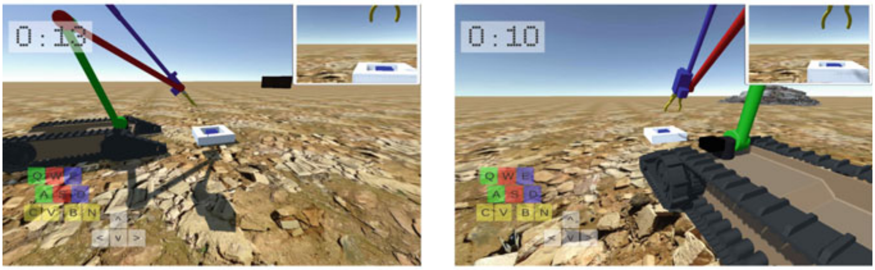
Our explicit path risk representation does not exclude traditional state-dependent risk elements. Those risk functions $r : s_i \mapsto ri$ include risk elements caused by the distance to the closest obstacles $r_{di} = dis(s_i)$, visibility from isovist lines (Fig. 4 upper left) $r_{vi} = vis(s_i)$, altitude due to propeller ceiling or ground effect $r_{ai} = alt(s_i)$, and tether singularity $r_{si} = sig(s_i)$. Those risk elements are additive along the path. Path-



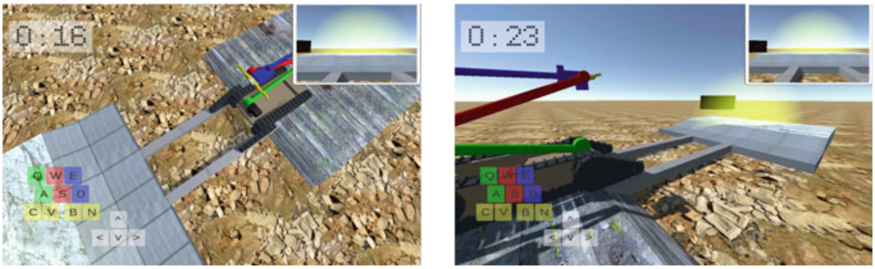
(a) Good Manipulability Viewpoint (b) Bad Manipulability Viewpoint



(c) Good Passability Viewpoint (d) Bad Passability Viewpoint



(e) Good Reachability Viewpoint (f) Bad Reachability Viewpoint



(g) Good Traversability Viewpoint (h) Bad Traversability Viewpoint

Fig. 3 Ongoing viewpoint quality study in simulation with professional PackBot operators

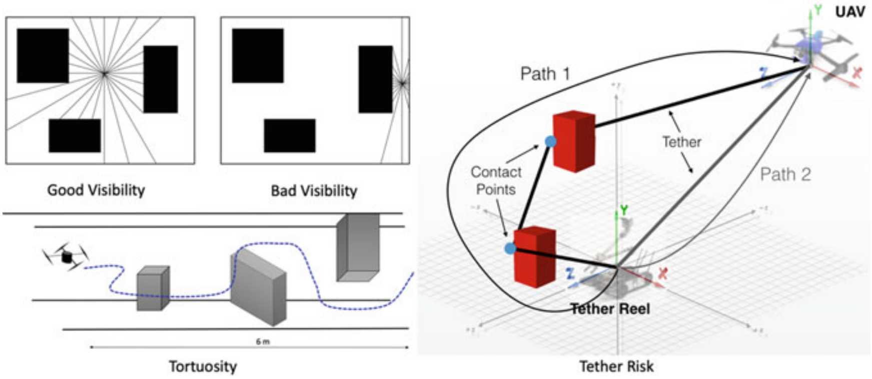


Fig. 4 Examples of risk elements: visibility (upper left), tortuosity (lower left), tether length, number of contacts, and azimuth (right)

dependent risk elements include action length, access element, tortuosity, number of tether contacts, tether length, and azimuth (Table 1). In order to define these risk elements, we further define the action between two consecutive states s_{i-1} and s_i to be A_i. So the whole sequence of actions to execute the entire path P is defined as $\mathbb{A} = \{A_1, A_2, ..., A_n\}$. For action length, $\|A_i\|$ denotes the length of the executing action A_i. For access elements, the function AE evaluates the difficulty of entering from the void where s_{i-1} is located to the void of s_i. The only positive difficulty is added to the risk index. Tortuosity characterizes the number of "turns" necessary to reach the state. More generally speaking, this is the difference by some measurement between two consecutive actions $\|A_i - A_{i-1}\|$. Tether length is a function of the entire path, e.g., taking path 1 and path 2 in Fig. 4 right will have a completely different tether length. The number of contact points and azimuth angle are also different. Risk index should never decrease with the execution of a path, which is guaranteed by the norm and max operations for the first three elements in Table 1. Thus, they only need to be evaluated once for each path (unitary). For the last three, however, the risk associated with each state may decrease, e.g., contact points may be relaxed [15] and tether length may decrease. But this does not cancel the previous risk. Therefore, those three elements need to be added to all states. Given a path P, its execution risk could be evaluated based on each risk element. Weighted sum or fuzzy logic could be used to combine all elements into one total risk index, quantifying the difficulty of executing that path. Detailed information on the explicit risk representation could be found in [13].

Table 1 Path-dependent risk elements

Risk element	Risk index	Property
Action length	$R_{AL}(P) = \sum_{i=1}^{n} \lVert A_i \rVert$	Unitary
Access element	$R_{AE}(P) =$ $\sum_{i=2}^{n} \max(AE(void(s_{i-1}), void(s_i)), 0)$	Unitary
Tortuosity	$R_T(P) = \sum_{i=2}^{n} \lVert A_i - A_{i-1} \rVert$	Unitary
Tether length	$R_{TL}(P) = f(s_0, s_1, ..., s_n)$	Additive
Number of contacts	$R_{NC}(P) = g(s_0, s_1, ..., s_n)$	Additive
Azimuth	$R_A(P) = h(s_0, s_1, ..., s_n)$	Additive

3.3 *Risk-Aware Reward-Maximizing Planning*

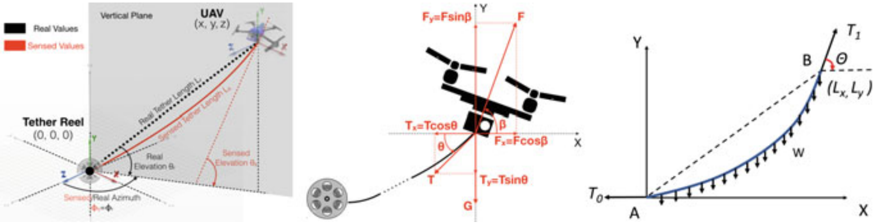
Given a viewpoint quality map as a reward and motion execution risk as a function of the path, the risk-aware reward-maximizing planner plans a minimum-risk path to each state [14], evaluates the reward collected, and then picks the one with optimal utility value, defined as the ratio between the reward and the risk. Executing the optimal utility path approximates the optimal visual assistance behavior.

4 Tethered Motion

With a high-level risk-aware optimal utility path, this section presents a low-level motion suite to realize the path on the tethered aerial visual assistant.

4.1 *Tether-Based Indoor Localization*

Our aerial visual assistant uses its tether to localize in GPS-denied environments. The sensory input is the tether length L, elevation angle θ, and azimuth angle ϕ. The mechanics model M in [17] corrects the preliminary localization model under taut and straight tether assumption (Fig. 5a) using the Free Body Diagram (FBD) of the UAV (Fig. 5b) and tether (Fig. 5c) in order to achieve accurate localization of the airframe $M : (\theta, \phi, L) \mapsto (x, y, z)$, from tether sensory input to Cartesian space location.


Real Values
Sensed Values
Vertical Plane
UAV
(x, y, z)
Tether Reel
(0, 0, 0)
Real Tether Length L
Sensed Tether Length L
Real Elevation θ
Sensed Elevation θ
Sensed/Real Azimuth
Y
X
Z
$F_y=F\sin\beta$
F
Y
$T_x=T\cos\theta$
β
$F_x=F\cos\beta$
X
θ
T
$T_y=T\sin\theta$
G
Y
T_1
B
Θ
(L_x, L_y)
W
T_0
A
X

(a) Localization Model (b) FBD of UAV (c) FBD ofTether

Fig. 5 Tether-based localization [17]

4.2 Motion Primitives

Two types of motion primitives are used, which map the waypoints in Cartesian space into tether-based motion commands: position control uses the inverse transformation from polar to Cartesian coordinates and three independent PID controllers to drive the position of L, θ, and ϕ to their desired values (Eq. 1). On the other hand, velocity control based on the system's inverse Jacobian matrix computes velocity commands L', θ', and ϕ' using an instantaneous velocity vector pointing from the current location to the next waypoint $d\overrightarrow{\mathbf{x}}/dt$ (Eq. 2). The vehicular yaw and camera pitch and roll reactively point at the center of the affordance along the entire path.

$$\begin{cases} L = & \sqrt{x^2 + y^2 + z^2} \\ \theta = arc\sin \frac{y}{\sqrt{x^2+y^2+z^2}} \\ \phi = & a\tan 2(\frac{x}{z}) \end{cases} \tag{1}$$

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{pmatrix} = \begin{pmatrix} \cos\theta\sin\phi & -L\sin\theta\sin\phi & L\cos\theta\cos\phi \\ \sin\theta & L\cos\theta & 0 \\ \cos\theta\cos\phi & -L\sin\theta\cos\phi & -L\cos\theta\sin\phi \end{pmatrix} \begin{pmatrix} L' \\ \theta' \\ \phi' \end{pmatrix} \tag{2}$$

The vehicular yaw and camera gimbal pitch are controlled using the 3-D vehicular position localization and the 3-D Cartesian coordinates of the visual assistance point of interest. The camera gimbal roll is passively controlled to align with gravity so that the operator's viewpoint is in level with the ground. Therefore, the visual assistant's camera is pointing toward the point of interest along the entire motion sequence [4, 11]. Xiao et al. [12] report detailed benchmarking results of the motion primitives.

4.3 Tether Contact Planning

In the case when some good viewpoints are located behind an obstacle and the UAV cannot reach with a straight tether, contact points of the tether with the environment

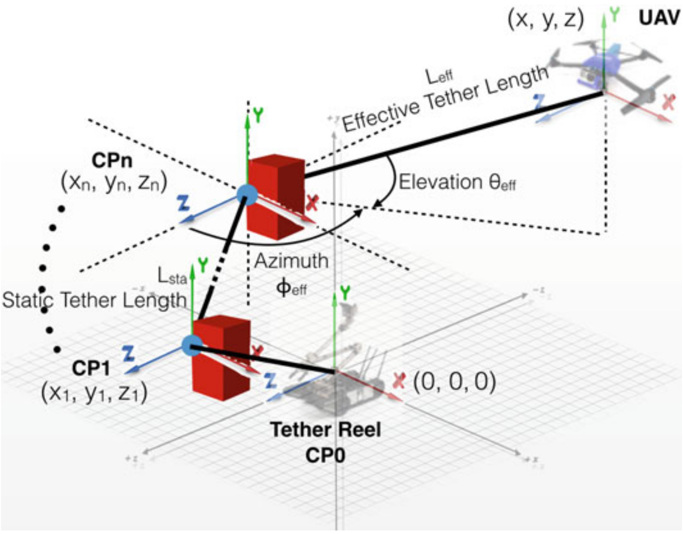

(x, y, z)
UAV
L_{eff}
Effective Tether Length
CPn
(x_n, y_n, z_n)
Elevation θ_{eff}
Azimuth
ϕ_{eff}
L_{sta}
Static Tether Length
CP1
(x_1, y_1, z_1)
(0, 0, 0)
Tether Reel
CP0
Y
Z

Fig. 6 Motion execution with contact(s) planning and relaxation: given multiple contact points along the tether, static tether length denotes the portion of the tether that wraps around the obstacles (Eq. 3), while the effective length is the last moving segment (Eq. 4). Effective elevation and azimuth angles (Eq. 4) are with respect to the last contact point (CP_n), instead of the tether reel (CP_0)

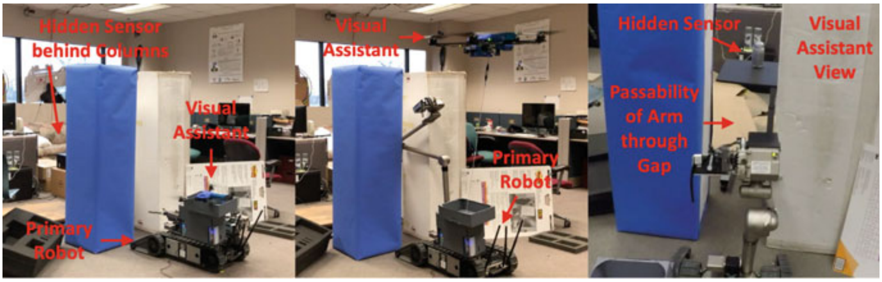
are necessary. The tether contact point(s) planning and relaxation framework in [15], which allows the UAV to fly as if it were tetherless, is implemented on the tethered visual assistant. A new contact is planned when the current contact is no longer within the line-of-sight of the UAV, while the current contact is relaxed when the last contact becomes visible again. Figure 6 shows the motion execution with multiple contact points (CPs).

$$L_{sta} = \sum_{0}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{3}$$

$$\begin{cases} L_{eff} = \sqrt{(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2} \\ \theta_{eff} = \quad arc\sin(\frac{y-y_n}{\sqrt{(x-x_n)^2+(y-y_n)^2+(z-z_n)^2}}) \\ \phi_{eff} = \quad a\tan 2(\frac{x-x_n}{z-z_n}) \end{cases} \tag{4}$$

5 System Demonstration

This section presents two system demonstrations in both indoor and outdoor environments and shows the enhanced situational awareness of the operator achieved by the visual assistance.



(a) Entering the Scene (b) Deploying for *Passability* (c) Visual Assistant View

Fig. 7 Visual assistance for *Passability*

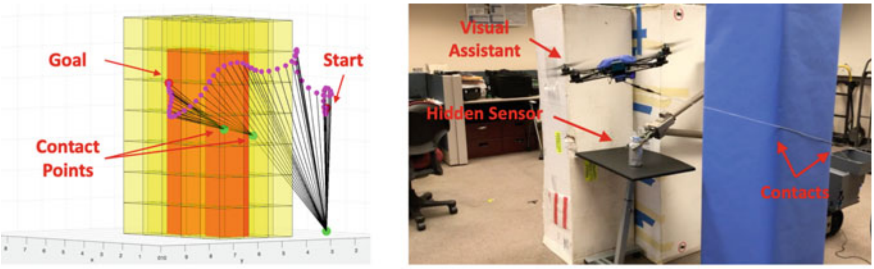
5.1 Indoor Test

In this demonstration, the co-robots team drives into a cluttered indoor environment, with the aim of retrieving a hidden sensor. The ground robot is tele-operated and creates a map of the environment. The entry points to the sensor are all blocked by the clutter, leaving the only retrieval option through the narrow gap between the two columns (shown in blue and white in Fig. 7a). Based on the viewpoint quality for *passability*, the visual assistant takes off and deploys to a viewpoint from behind and above to help perceive arm *passability* through a gap (Fig. 7b). The visual assistant view is shown in Fig. 7c, where the relative location of the arm to the narrow gap along with the hidden sensor is well perceived.

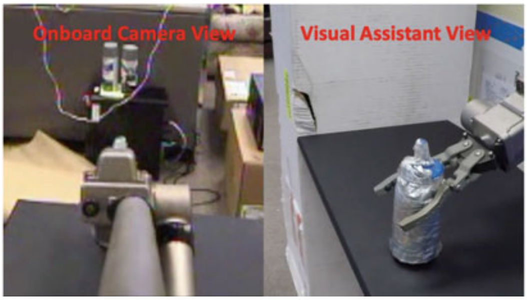
After the arm passes through the gap, the visual assistant switches to assist with *manipulability*. Good viewpoints for *manipulability* are located at the side of the gripper. After balancing the viewpoint quality reward and motion execution risk, the planner finds a goal and a path leading to it, which contains two tether contact points with the obstacles. Figure 8a shows the obstacles (red), inflated space for UAV flight tolerance (yellow), waypoints on the planned path (purple), and two contact points on the obstacles (green). The tether configuration is illustrated with black lines. The actual deployment is shown in Fig. 8b. The onboard camera view on the left of Fig. 8c completely misses the depth perception. With this onboard view alone, the risk of not reaching or even knocking off the sensor is high. This lack of depth perception is compensated by the visual assistant view (right).

5.2 Outdoor Test

The co-robots team is also deployed in an outdoor disaster environment, Disaster City® Prop 133 in College Station, Texas (Fig. 9). The environment simulates a collapsed multi-story building and the mission for the co-robots team is to navigate into the building and search for victims and threats in two stranded cars.

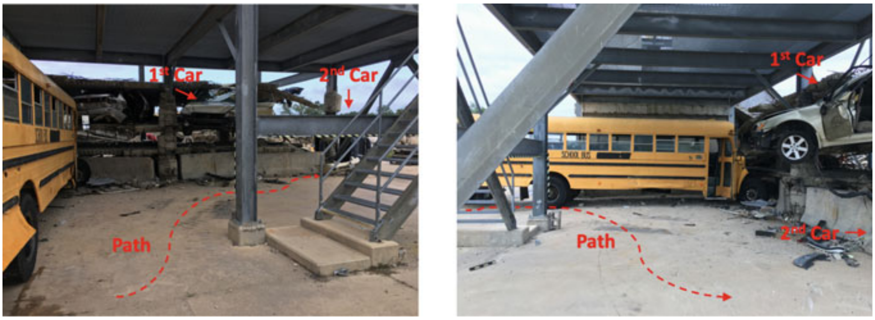

Goal
Start
Contact Points
Visual Assistant
Hidden Sensor
Contacts

(a) Path Planning with 2 Contacts (b) Deploying for *Manipulability*


Onboard Camera View
Visual Assistant View

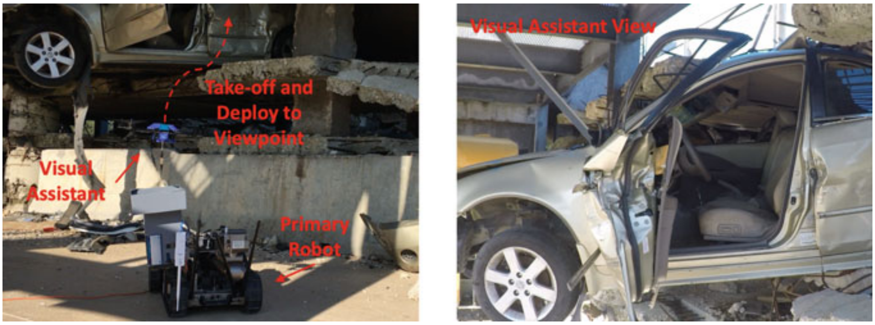
(c) Onboard Camera and Visual Assistant View

Fig. 8 Visual assistance for *Manipulability*


1st Car
2nd Car
Path
1st Car
Path
2nd Car

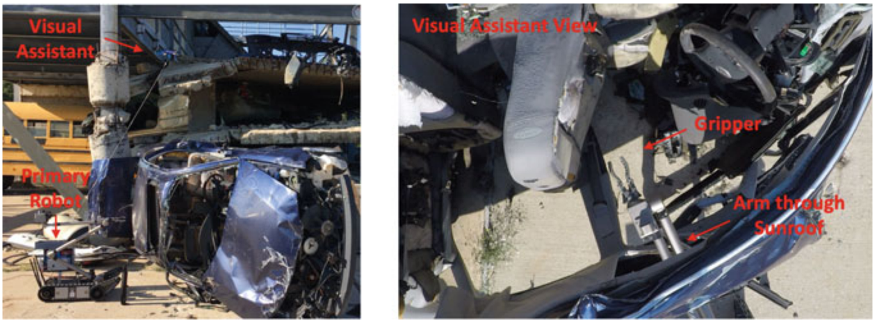
(a) View from Entry Point (b) View from End Point

Fig. 9 Disaster City ® Prop 133


Take-off and Deploy to Viewpoint
Visual Assistant
Primary Robot
Visual Assistant View

(a) Take-off and Deployment (b) No Victim in 1st Car

Fig. 10 Enhanced coverage through visual assistance for the first car


Visual Assistant
Primary Robot
Visual Assistant View
Gripper
Arm through Sunroof

(a) Inspection for 2nd Car (b) Assisting Insertion Depth Perception

Fig. 11 Car inspection through open sunroof for the second car

After reaching the first stranded car, which was on the second floor but is now squeezed down by the collapse, the primary robot's onboard camera is not able to reach the height to search for victims inside the car. The visual assistant takes off from the landing platform and autonomously navigates to a manually specified viewpoint to look inside the car (Fig. 10a). Through the elevated viewpoint provided by the visual assistant, it is confirmed that no victim is trapped in the first car. The visual assistant lands back on the primary robot and the team is tele-operated to the second car.

The second car was tipped over during the collapse, with its sunroof open on the side. The operator intends to insert the manipulator arm into the interior for a thorough search and retrieval if necessary. For safety reasons, the goal is not automatically selected, but manually specified above the side window (Fig. 11a). Looking down through the side window, the depth of the arm insertion into the car interior is clearly visible. No victim or hazardous material exists in the car. The visual assistant lands, the co-robots team finishes the mission and navigates back.

6 Conclusions

We present a co-robots team equipped with autonomous visual assistance for robot tele-operations in unstructured or confined environments using a tethered UAV. The tele-operated primary ground robot projects human presence to remote environments, while the autonomous visual assistant provides enhanced situational awareness to the human operator. The autonomy is realized through a formal study on viewpoint quality, an explicit risk representation to quantify the difficulty of path execution, and a planner that balances the trade-off between viewpoint quality reward and motion execution risk. With the help of a low-level motion suite, including tether-based localization, motion primitives, and contact(s) planning, the high-level path is implemented on a tethered aerial visual assistant given the existence of obstacles. The co-robots team is deployed in both indoor and outdoor search and rescue scenarios, as a proof of the concept of the system. Future work will focus on quantitatively measuring the performance of the co-robots team, including the reward collected, risk encountered, flight accuracy of the autonomous visual assistant, and the improvement in the tele-operation of the primary robot.

Acknowledgements This work is supported by NSF 1637955, NRI: A Collaborative Visual Assistant for Robot Operations in Unstructured or Confined Environments.

References

1. Chaimowicz, L., Cowley, A., Gomez-Ibanez, D., Grocholsky, B., Hsieh, M., Hsu, H., Keller, J., Kumar, V., Swaminathan, R., Taylor, C.: Deploying air-ground multi-robot teams in urban environments. In: Multi-Robot Systems. From Swarms to Intelligent Automata, vol. III, pp. 223–234. Springer (2005)
2. Chaimowicz, L., Grocholsky, B., Keller, J.F., Kumar, V., Taylor, C.J.: Experiments in multirobot air-ground coordination. In: IEEE International Conference on Robotics and Automation, 2004, Proceedings, ICRA'04, vol. 4, pp. 4053–4058. IEEE (2004)
3. Cheung, C., Grocholsky, B.: UAV-UGV collaboration with a PackBot UGV and Raven SUAV for pursuit and tracking of a dynamic target. In: Unmanned systems technology X, vol. 6962, p. 696216. International Society for Optics and Photonics (2008)
4. Dufek, J., Xiao, X., Murphy, R.: Visual pose stabilization of tethered small unmanned aerial system to assist drowning victim recovery. In: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pp. 116–122. IEEE (2017)
5. Frietsch, N., Meister, O., Schlaile, C., Trommer, G.: Teaming of an UGV with a VTOL-UAV in urban environments. In: 2008 IEEE/ION Position, Location and Navigation Symposium, pp. 1278–1285. IEEE (2008)
6. Kiribayashi, S., Yakushigawa, K., Nagatani, K.: Design and development of tether-powered multirotor micro unmanned aerial vehicle system for remote-controlled construction machine. In: Field and Service Robotics, pp. 637–648. Springer (2018)
7. Murphy, R., Dufek, J., Sarmiento, T., Wilde, G., Xiao, X., Braun, J., Mullen, L., Smith, R., Allred, S., Adams, J., et al.: Two case studies and gaps analysis of flood assessment for emergency management with small unmanned aerial systems. In: 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 54–61. IEEE (2016)

8. Murphy, T.B.: Apprehending remote affordances: assessing human sensor systems and their ability to understand a distant environment. Ph.D. Thesis, The Ohio State University (2013)
9. Rao, R., Kumar, V., Taylor, C.: Visual servoing of a UGV from a UAV using differential flatness. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453), vol. 1, pp. 743–748. IEEE (2003)
10. Xiao, X., Cappo, E., Zhen, W., Dai, J., Sun, K., Gong, C., Travers, M.J., Choset, H.: Locomotive reduction for snake robots. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3735–3740. IEEE (2015)
11. Xiao, X., Dufek, J., Murphy, R.: Visual servoing for teleoperation using a tethered UAV. In: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pp. 147–152. IEEE (2017)
12. Xiao, X., Dufek, J., Murphy, R.: Benchmarking tether-based UAV motion primitives. In: 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE (2019)
13. Xiao, X., Dufek, J., Murphy, R.: Explicit motion risk representation. In: 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE (2019)
14. Xiao, X., Dufek, J., Murphy, R.: Explicit-risk-aware path planning with reward maximization (2019). arXiv:1903.03187
15. Xiao, X., Dufek, J., Suhail, M., Murphy, R.: Motion planning for a UAV with a straight or kinked tether. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8486–8492. IEEE (2018)
16. Xiao, X., Dufek, J., Woodbury, T., Murphy, R.: UAV assisted USV visual navigation for marine mass casualty incident response. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6105–6110. IEEE (2017)
17. Xiao, X., Fan, Y., Dufek, J., Murphy, R.: Indoor UAV localization using a tether. In: 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 1–6. IEEE (2018)

Tethered Dual-Wheeled Robot for Exploration of Volcanic Fumaroles on Steep Slopes

Keiji Nagatani and Masaki Momii

1 Introduction

Japan is a volcanic country with 111 active volcanoes, which is about 7% of all the active volcanoes in the world. Volcanic eruptions cause serious damage to the surrounding environments and societies, and consideration of evacuation methods based on eruption levels is essential. The Japan Meteorological Agency has designated 50 out of 111 active volcanoes as constantly observed volcanoes that require a thorough monitoring and observation system for volcano disaster prevention. In these volcanoes, volcano observation facilities, such as seismographs, inclinometers, air vibrometers, GNSS observation devices, surveillance cameras have been installed to capture the precursors of the eruption and appropriately issue eruption alerts. Such monitoring is conducted 24 h a day.

On the other hand, periodic manned volcano observation is also performed by the Meteorological Agency and academic institutions. Volcanic gas is highly mobile and quickly transmits underground (magma) information to the surface. Therefore, analysis of the chemical composition and measurement of the emission amount of the volcanic gas can act as clues that reveal the status of the magma and volcano. However, because volcanic gas is typically of high temperature and contains toxic chemical components, the manned observation of volcanic fumaroles is dangerous. Therefore, the development of a robot that substitutes humans for the performance of these tasks is expected. Additionally, some fumaroles exist in depressions and steep slopes, thus, it is difficult for typical wheeled robots and tracked vehicles to reach them. Therefore, in the ex-research, the authors' group has developed a

---

K. Nagatani (✉)
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: keiji@ieee.org

M. Momii
Tohoku University, 6-6-10, Aramaki-Aoba, Sendai, Japan

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_3

lightweight four-wheeled mobile robot with a tether to investigate volcanic fumaroles on steep slopes [1]. The robot performed well on steep slope traversal using a tether. However, in field tests in volcanic environments, the robot occasionally tipped over. As a result, it has been found that the operation of such robots in real volcanic environments is difficult. Robots, for this purpose, are required to have a more reliable and robust locomotion while maintaining their lightness. In this research, to meet the above requirements, a steep-slope mobile robot is proposed for the investigation of fumaroles. Additionally, experimental results in volcanic fields proved the advantages of the proposed robot.

2 Related Work

In related works, tethered mobile robots have been used to traverse a steep slope. Dante and Dante II, developed by Carnegie Mellon University, were early tethered robots aimed at moving in extreme environments with eight legs. In 1994, Dante II succeeded in surveying the crater of a snow-covered volcano [2]. TRESSA, made by NASA JPL, was a four-wheeled mobile robot that used two tethers and achieved more stable traversability than one using a single tether [3]. However, some problems were reported. Because two tethers were used, contact between the tethers occurred on the slope, which caused the unstable motion of the robot. Furthermore, the tether encountered the ground and wore out because a tether pulling device was not built into the robot. Axel rover, also developed by NASA JPL, was a dual-wheeled mobile robot with a built-in tether-traction device that performed excellently when it was made to traverse rough slopes. [4]. Furthermore, DuAxel rover system, which connected two Axel rovers serially, was a tethered robot system that did not require the installation of anchors. Instead, one Axel rover acted as an anchor for the other rover.

However, because the direction of the rover was restricted by tether pulling, the authors conclude that the turning motion on a slope was difficult, and that its moving range was limited. The system of Moonraker and Tetris, developed by Yoshida's lab, was based on the same concept [5].

VolcanoBot, developed by NASA JPL, was a tether-traction type dual-wheeled mobile robot for mapping craters. The robot could be controlled by teleoperation using the connected tether, and succeeded in observing the insides of crater crevices that humans cannot enter. TReX, developed by the University of Toronto, was a tethered four-wheeled mobile robot aimed at mapping steep slope environments [6]. Because this robot could arbitrarily change its traversal direction independent of its tether direction, it had a wider range of movement on the slope in comparison to other robots, such as legged or dual-wheeled robots. However, because the number of wheels is more than that for others, the wheels' diameter should be smaller than other robots, and its traversability on rough terrain tends to be reduced.

In this study, the authors focus on mechanisms of dual-wheeled robots, such as Axel rovers, to retain traversability on rough terrains.

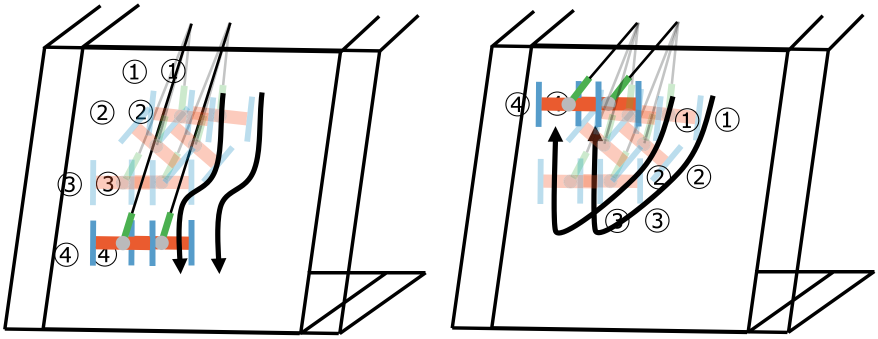
3 System Design of a Steep Slope Traversal Robot

3.1 *Proposal of Passive Tether Guide Module*

This research focuses on the dual-wheeled robot. It has better traversal performance than others because it can mount large wheels. One problem is that its steering performance is limited on a slope. In this research, the authors propose an improved version of the dual-wheeled robot to solve this problem. A feature of this robot is that it has a passive tether-guide module to extend its turning range on a slope. The module, which includes a tether-guide and tether-traction mechanism, is connected passively to the robot. This improves its turning motion, diagonal traversal motion, and crossing motion on a slope as shown in Fig. 1. Additionally, as shown in Fig. 2, the proposed mechanism exhibits good turning motion performance when compared to a four-wheeled robot (TReX), and it traverses rough terrain with performance similar to that of a single-tether dual-wheeled robot (Axel). In the following subsections, the proposed steep slope traversal method and development of a prototype are discussed.

3.2 *Steep Slope Traversal Method*

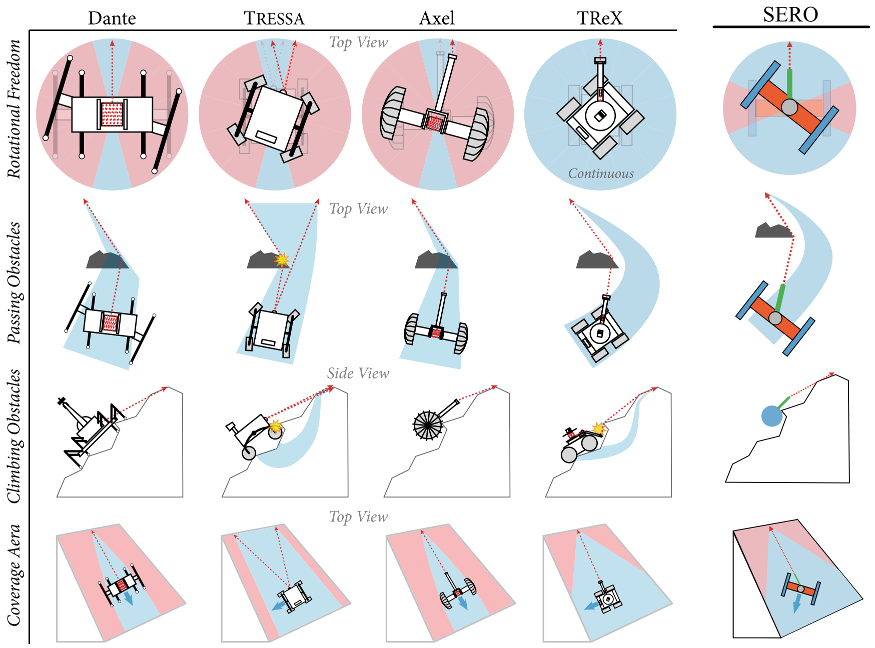
When a robot moves to an arbitrary goal point on a slope, it is necessary to extend its tether length, while keeping its front directed at the goal point and simultaneously rotating its wheels so that it does not slip. To realize this motion, it is necessary to coordinate the control of the wheels and the tether-traction mechanism. If the wheel rotation does not match the tether delivery, the robot may slip and tip over, causing failure to reach the goal position. In addition, both mechanisms may provide a large load to each other, and the robot itself may be damaged. In this research, to satisfy the above requirement, speed control is applied to the tether-traction mechanism, and torque control is applied to the driving wheels. By generating a constant tension T_{bias}



(a) Traversing / diagonal motion

(b) Traversing motion

Fig. 1 Method using tether-guide rotation mechanism


Dante
TRESSA
Axel
TReX
SERO
Rotational Freedom
Passing Obstacles
Climbing Obstacles
Coverage Aera
Top View
Continuous
Top View
Side View
Top View

(a) Conventional tethered mobile robots

(b) Tethered dual-wheel mobile robot with a passive tether guide module.

Fig. 2 Comparison among tethered mobile robots. The **a** part is quoted from the literature [7], and the **b** part is the authors' proposal

from the driving wheels to the tether tension, the wheels rotate while maintaining tether tension when the tether is put in and out. Thus, coordinated control of both the wheels and the tether-traction mechanism is realized. In addition, to realize stable steering control of the robot on rough terrains, the control amount T_{diff}, calculated by PD controller for azimuth angle is introduced. By using T_{bias} and T_{diff}, the reference torque of the wheels (T_R, T_L) are determined using the following equations:

$$T_R = T_{\text{bias}} + T_{\text{diff}} \tag{1}$$

$$T_L = T_{\text{bias}} - T_{\text{diff}}, \tag{2}$$

where

$$T_{\text{diff}} = K_p e(t) + K_d \frac{de(t)}{dt} \tag{3}$$

$$e(t) = \psi_{\text{measured}} - \psi_{\text{target}}. \tag{4}$$

(see Fig. 3 also.)

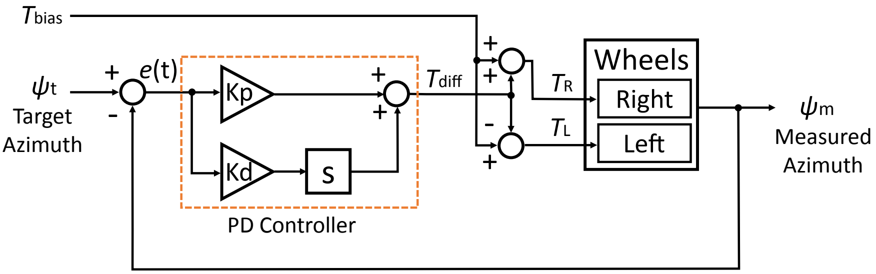

T_{bias}
ψ_t
Target
Azimuth
+
-
$e(t)$
Kp
Kd
s
+
+
PD Controller
T_{diff}
+
+
-
+
T_R
T_L
Wheels
Right
Left
ψ_m
Measured
Azimuth

Fig. 3 Wheel torque controller

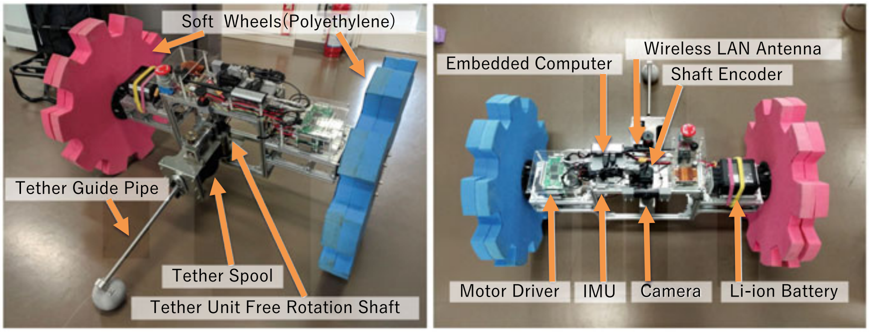

Soft Wheels(Polyethylene)
Tether Guide Pipe
Tether Spool
Tether Unit Free Rotation Shaft
Embedded Computer
Wireless LAN Antenna
Shaft Encoder
Motor Driver
IMU
Camera
Li-ion Battery

Fig. 4 An overview of SEROW

3.3 *Development of Steep Slope Exploration RObot with Dual Wheels*

Based on the previous subsections, the authors developed a prototype of the dual-wheeled robot for steep slope traversal, called Steep slope Exploration RObot with dual Wheels (SEROW), as shown in Figs. 4 and 5. The specification of the robot is presented in Table 1. The tether-guide module includes a winch mechanism. By connecting the module to the body via bearings, free rotation of the module is realized. The guide module can rotate ±50°, and the winch is integrated into the module. The total length of the tether is 20 m. Additionally, the winch drives the tether through a worm gear. With the self-locking function of the worm mechanism, when the motor of the robot is not driven, it can keep its position on the slope without consuming electric power.

Soft wheels made of polyethylene with a diameter of 500 mm and width of 60 mm are used as for the robot to enhance its traversability on rough terrain. Each wheel also mounts ten grousers whose depths are 50 mm each. In addition, by designing the distance between its tires to be wider than the shoulder width of a typical human, it is easy to carry in a backpack. The robot weighs a total of 13 kg and is light enough for humans to carry across long distances.

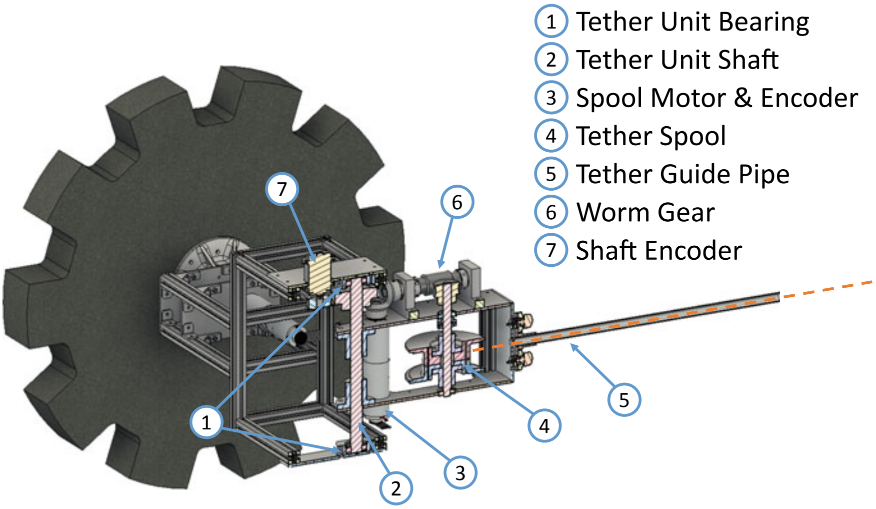

① Tether Unit Bearing
② Tether Unit Shaft
③ Spool Motor & Encoder
④ Tether Spool
⑤ Tether Guide Pipe
⑥ Worm Gear
⑦ Shaft Encoder
7
6
1
5
4
3
2

Fig. 5 Cut view of SEROW

Table 1 System specifications

Dimension	750 × 830 × 500 mm (L × W × H)
Mass	13 kg
Power	Li-ion 23.1V 5.5Ah
Computer	Raspberry Pi 3 B
IMU	MPU9250 (RT-USB-9axisIMU2)
Motor driver	T-frog TF-2MD3-R6
Wheel motor	Maxon DCX26L GB KL18V
Winch motor	Maxon DCX32L GB KL 18V
Tether	ϕ 1 mm 30 m SUS304

3.4 Evaluation of the Passive Tether-Guide Module

An evaluation test was conducted to confirm whether the tethered dual-wheeled mobile robot improved its turning capability on steep slopes with the use of its passive tether-guide module. For this test, simulated slopes made of veneers with variable inclination were used. The tether was fixed to the upper part of the slope via a pulley. To evaluate the usefulness of the passive tether-guide module, two conditions were set: (1) the module was fixed and, (2) the module moved passively. The experimental procedure of the test is as follows:

1. Set the robot horizontally on the slope,
2. Apply reverse torque to the right and left wheels, and then measure the turning angle of the robot, continuously.

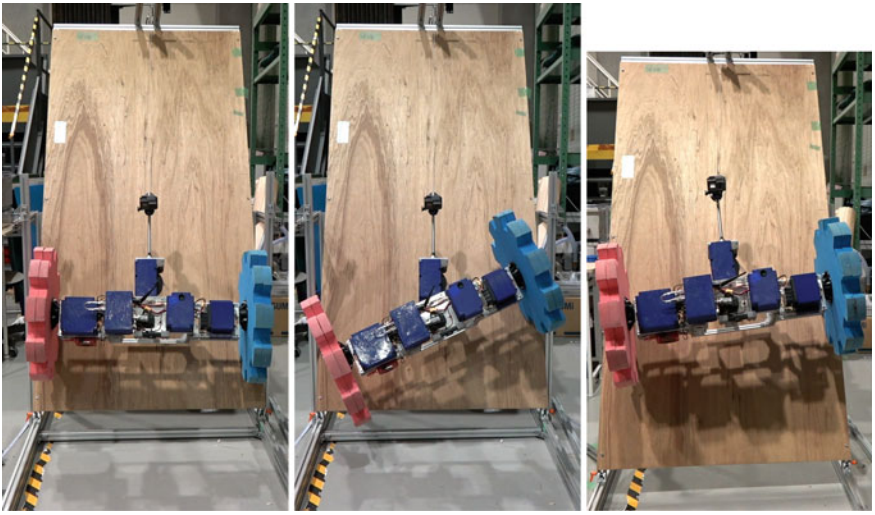
3. Increase the torque until the wheels slip occurs.
4. Conduct each measurement test and increase the slope angle from 20° to 80° in 20° increments.

3.5 Results of the Experimental Test

Table 2 presents the comparison result based on the presence of the passive module. Additionally, Fig. 6 shows the experimental scenes for the case when the slope angle is 60° and absolute value of the torque is 15 Nm. From the results, it is seen that when the passive tether-guide module is a passive connection, the robot turns more than 40° in all tests. On the other hand, when the tether-guide module was fixed to

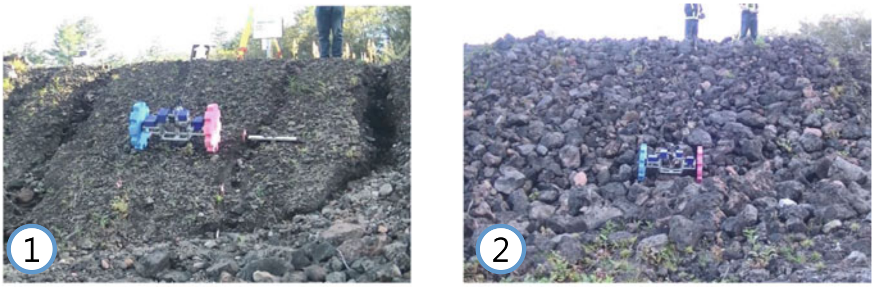
Table 2 Comparison result based on the presence of the passive module

Slope angle [°]	Maximum steering angle [°]	
	Passive module	Fixed module
20	Over 40	33.1
40	Over 40	14.4
60	Over 40	8.4
80	Over 40	4.5



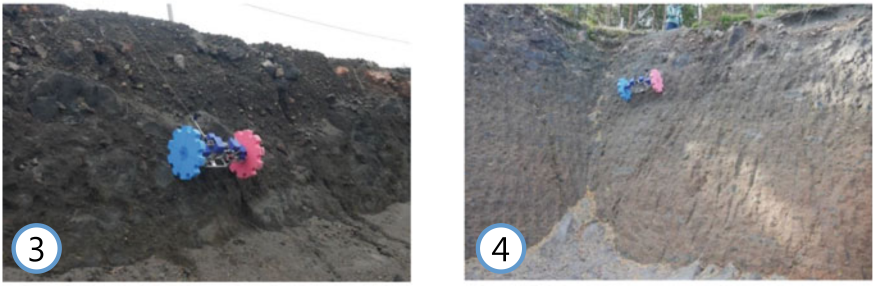
(a) Initial pose (b) The module was in passive (c) The module was fixed.

Fig. 6 Experimental scenes based on the presence of the passive module when the slope angle is 60° and absolute value of the wheel torque is 15 Nm



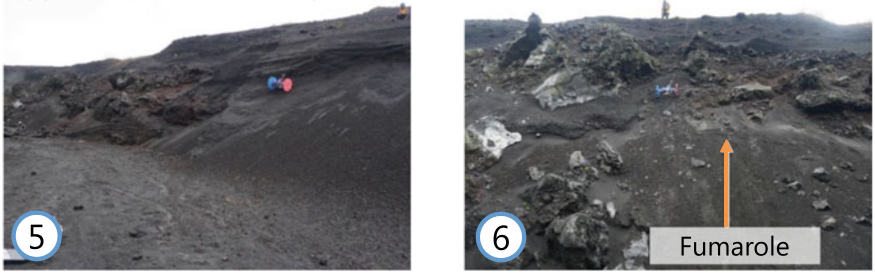
Some plants and gravels (5cm maximum) are on the slope

Huge rocks are on the slope



Lower half is pyroclastic flow deposit, upper half is sediment deposited

Pyroclastic flow deposit



Scoria

Scoria, huge rocks, gravels

Fig. 7 Target fields

the robot, it did not succeed in achieving a turning motion over 40°. Furthermore, the maximum turning angle decreased with an increase in slope inclination. Based on the above, it was confirmed that the passive tether-guide module improves the turning capability of the robot on steep slopes.

4 Field Experiments

4.1 Steep Slope Traversal

To evaluate the traversal performance of the robot on steep slopes, field experiments were conducted on 6 different kinds of slopes, as shown in Fig. 7. In these field experiments, the robot was made to go up and down different steep slopes and we checked whether it succeeded or not.

The procedure for all field experiments is as follows.

1. First, an anchor is installed at the top of the slope, and the end of the tether is fixed with a carabiner.
2. The operator controls the robot from a position where the robot can be visually recognized.
3. The operator descends the robot to the deepest part of the slope while avoiding obstacles on the slope.
4. The operator ascends the robot to the initial position.

The test environment and test results for each field are presented in Table 3. The results indicate that the robot succeeded in traversing all slopes. However, for a small number of cases, a failure of wireless communication may have occurred, resulting in situations where the robot could not be controlled. Failures in Exp. 3 and Exp. 6, as seen in Table 3, were caused by the failure of wireless communication.

4.2 Experiment of Observation Scenario

In the Izu-Oshima field, a scenario experiment, the simulated investigation of a volcanic fumarole, was conducted. The target field includes some volcanic fumaroles, thus, some sensors, such as a gas detection sensor and temperature sensor, were mounted on the robot to obtain environmental information through teleoperation. The sensors mounted on the robot are shown in Fig. 8.

Table 3 Field experiments

	Location	Date	Average angle of slope [°]	Number of successes/attempts
Exp. 1	Asama-Yama	Oct. 2018	30	3/3
Exp. 2	Asama-Yama	Oct. 2018	30	2/2
Exp. 3	Asama-Yama	Oct. 2018	70	2/3
Exp. 4	Asama-Yama	Oct. 2018	70	3/3
Exp. 5	Izu-Oshima	Nov. 2018	40–70	2/2
Exp. 6	Izu-Oshima	Nov. 2018	40–70	2/3

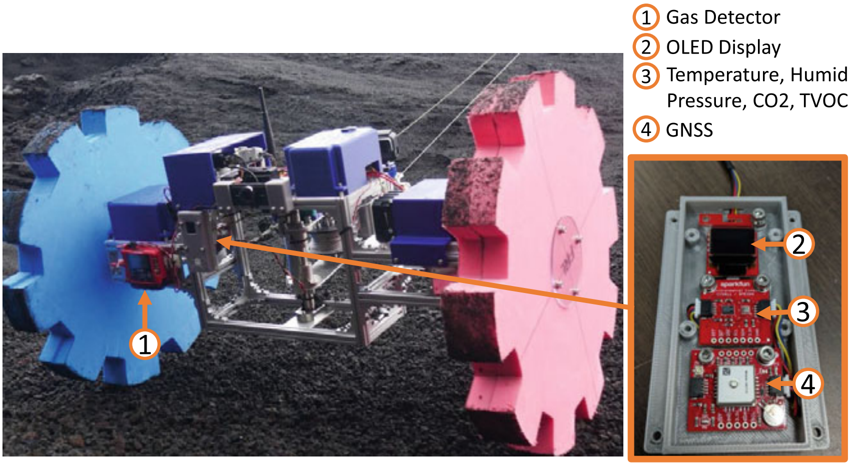


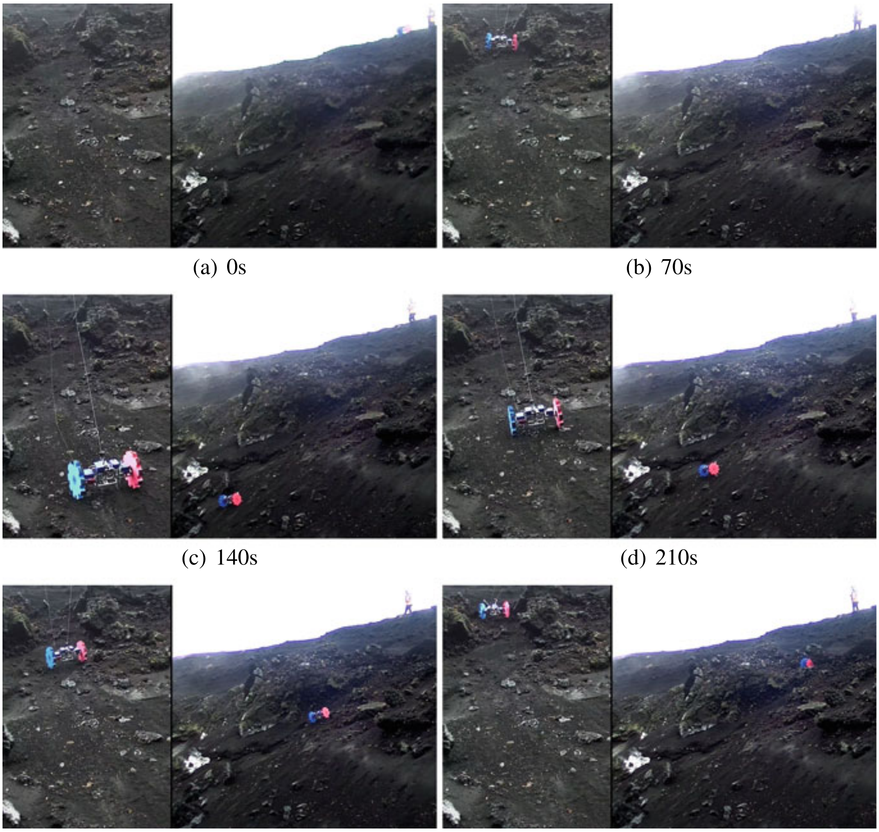
Fig. 8 Sensor configuration on the target robot

As a result of the experiment, the robot successfully approached the vicinity of the fumarolic pores, as shown in Fig. 9. Unfortunately, it failed to directly measure the gas through its gas-sensor.

4.3 Lessons Learned

In this section, the lessons learned from the field experiment are discussed.

1. Portability: For this experiment, one of the authors carried the robot and other experimental materials, which totalled to over 20 kg. The elevation difference was 100 m and total distance was 4 km. The total weight should be reduced in actual surveys. Additionally, the proposed robot has a shape in which a tether-guide pipe protrudes significantly from the robot body; therefore, it is dangerous for a person to carry it. Hence, such projections need to be designed to be easily removable, and SEROW has such a design.
2. Tether: There was no breakage of the tether throughout the experiment. However, there was some damage. Particularly, the tether near the anchor was strongly rubbed on the ground and sometimes damaged. Soils in volcanic environments are often sharp and prone to damaging the tether. Therefore, the resistance to cutting is as important as the load capacity when selecting a tether.
3. Communication: In some tests, communication failure occurred because the wireless LAN access point was installed in the wrong position. As a result, it was necessary to manually retrieve robots that could not be maneuvered on the slope in the middle of testing. To prevent such problems, for a tethered robot, communication speed and stability can be improved by using the tether itself



(a) 0s

(b) 70s

(c) 140s

(d) 210s

(e) 280s

(f) 350s

Fig. 9 Experimental scene of observation scenario

as a wired communication cable. In the current implementation, however, the metal tether was used to make it thin as much as possible. In addition, it requires to prepare a device that forcibly winds up the tether and recovers the robot if it becomes impossible to maneuver.

4. Controllability: In all experiments, the operator maneuvered the robot from a position where the robot could be seen directly. This was because it was difficult to maneuver with a wide-angle camera (view angle 120°) mounted in front of the robot. However, for actual operations, this problem must be solved because it is necessary to operate in a state where the robot cannot be seen. For example, laser-range-sensor-based mapping, a camera system installed at the end of the telescopic rod, or video streaming captured from an unmanned aerial vehicle are considered.
5. Passive tether-guide module: One of the features of the SEROW's mechanism is a passive tether-guide module. It enables the improvement of turning motion

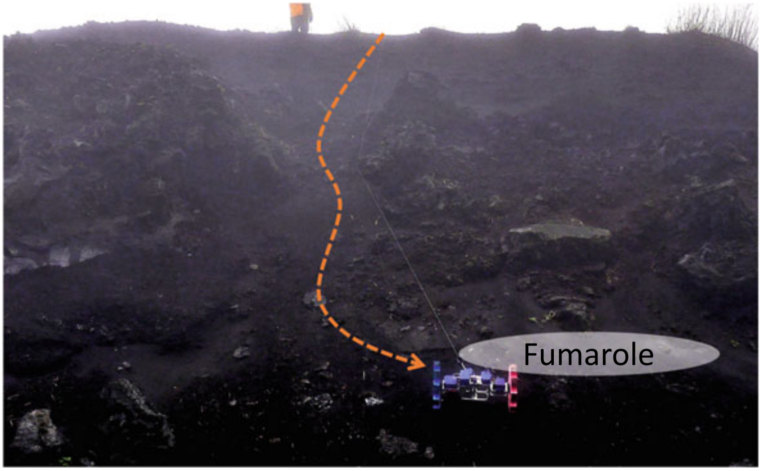


Fig. 10 Down-and-up-motion to move the robot horizontally. It proved the effectiveness of the Passive tether-guide module in a real environment

of the tethered robot. Actually, in experiment 6, observation scenario, the robot finally approached the vicinity of the fumarolic pores as shown in Fig. 10. It was required to go up and down to move horizontally, as shown in Fig. 1b. In case that it had no passive joint, it never succeeded in such a motion. On the other hand, it was thought that the passive joint might hind the traversal motion, particularly simple returning motion on the horizontal ground. Actually, in the beginning, one actuator was mounted at the passive joint part to fix it in case of unnecessary of passiveness. Practically, when the robot traced to the returning path straightly, the tether was always rewound, the passive joint was straight, and it never hind the desired motion. Finally, the actuator to fix the passive joint was never activated in these experiments.

6. Gas measurement: In the final test, the developed robot successfully approached the fumaroles. However, because the sensor location was not optimized, it failed to directly measure the gas. To complete gas measurement, an additional manipulation mechanism to insert the sensor head into the fumaroles may be required.

5 Conclusions

This study aimed to develop a lightweight mobile robot for the safe investigation of volcanic fumaroles located on steep slopes. To realize the above objective, a novel dual-wheeled tethered mobile robot was proposed, which was mounted with a tether-guide rotation mechanism to enable steering motion on a steep slope. It also enabled us to change the traversal direction of the robot independent of its tether direction, which made it possible to achieve turning motion, diagonal traversal motion, and crossing motion on a steep slope. Field experiments in real volcanic environments confirmed the validity and limitations of the proposed robot. The robot can traverse up to 70° slope composed of pyroclastic flow deposits, sediment deposits, or scoria. Additionally, it can conduct a survey of volcanic fumaroles by mounting sensors. In case the wireless communication between the robot and operator PC becomes unstable due to the cliff shield on the steep slope, control of the robot also becomes unstable. Therefore, a more stable communication system is required.

References

1. Nagatani, K., Tatano, S., Ikeda, K., Watanabe, A., Kuri, M.: Design and development of a tethered mobile robot to traverse on steep slope based on an analysis of its slippage and turnover. In: IROS, pp. 2637–2642 (2017)
2. Bares, J.E., Wettergreen, D.S.: Dante ii: technical description, results, and lessons learned. Int. J. Robot. Res. **18**(7), 621–649 (1999)
3. Huntsberger, T., Stroupe, A., Aghazarian, H., Garrett, M., Younse, P., Powell, M.: Tressa: teamed robots for exploration and science on steep areas. J. Field Robot. **24**(11–12), 1015–1031 (2007)
4. Nesnas, I.A.D., Matthews, J.B., AbadManterola, P., Burdick, J.W., Edlund, J.A., Morrison, J.C., Peters, R.D., Tanner, M.M., Miyake, R.N., Solish, B.S., et al.: Axel and duaxel rovers for the sustainable exploration of extreme terrains. J. Field Robot. **29**(4), 663–685 (2012)
5. Britton, N., Yoshida, K., Walker, J., Nagatani, K., Taylor, G., Dauphin, L.: Lunar micro rover design for exploration through virtual reality tele-operation. Field Service Robot. 259–272 (2015)
6. McGarey, P., Yoon, D., Tang, T., Pomerleau, F., Barfoot, T.D.: Developing and deploying a tethered robot to map extremely steep terrain. J. Field Robot. **35**(8), 1327–1341 (2018)
7. McGarey, P., Pomerleau, F., Barfoot, T.D.: System design of a tethered robotic explorer (TReX) for 3D mapping of steep terrain and harsh environments. In: Wettergreen, D., Barfoot, T.D. (eds.) Proceedings of the International Conference on Field and Service Robotics (FSR), Springer Tracts in Advanced Robotics, vol. 113, pp. 267–281. Toronto, Canada, 24–26 June 2015

Robotic Reliability Engineering: Experience from Long-Term TritonBot Development

Shengye Wang, Xiao Liu, Jishen Zhao, and Henrik I. Christensen

1 Introduction

While researchers continue to invent technology that extends robots' abilities to sense and interact with the environment, one of the often neglected research areas in robotics is reliability engineering. Robots often perform perfect demos under controlled settings and close human supervision, but they tend to be less reliable when working autonomously for an extended period in unstructured environments. The reliability issues have become an obstacle that prevents the robot from assisting people in their everyday lives. Unfortunately, the lack of resilience rooted in the research and prototype robot system development—many developers tend to find the simplest method to make the robot "just to work"—and the ad hoc solutions usually lead to a fragile system that has difficulties in feature iterations and failure analysis. The absence of "reliability thinking" drives the robots away from long-term autonomy, in which the robot continues to work and evolve over an extended period.

We built TritonBot to study reliability challenges in developing and deploying a long-term autonomous service robot that interacts with people in an open environment. TritonBot (Fig. 1) is a robot receptionist and a tour guide deployed in an office building at UC San Diego. It greets visitors, recognizes visitors' faces, remembers visitors' names, talks to visitors, and shows visitors the labs and facilities in the

---

S. Wang (✉) · X. Liu · J. Zhao · H. I. Christensen
Department of Computer Science and Engineering, University of California San Diego, 9500 Gilman Drive, MC 0404, La Jolla, CA 92093, USA
e-mail: shengye@ucsd.edu

X. Liu
e-mail: x1liu@ucsd.edu

J. Zhao
e-mail: jzhao@ucsd.edu

H. I. Christensen
e-mail: hichristensen@ucsd.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_4

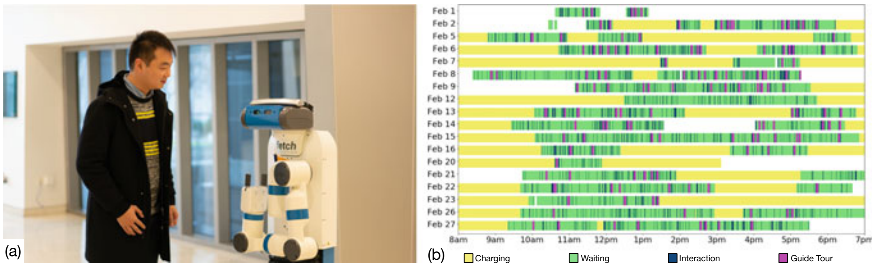


Fig. 1 **a** A visitor interacting with TritonBot. TritonBot can talk to people and show people around. **b** A one-month deployment log of TritonBot [24]. In February 2018, TritonBot worked (waiting, talking, and guiding tours) for 108.7 hours, and it actively interacted (talking and guiding tours) with people for 22.1 h

building. Our previous work [24] summarized lessons in the initial TritonBot development; this paper presents our efforts in making TritonBot more reliable during its long-term deployment.

Long-term autonomy consists of three primary factors: scalability, resilience, and learning: Scalability enables the robot system to grow and gain more features smoothly. Resilience allows the robot to adapt to environmental changes and tolerate transient errors. Learning helps the robot to benefit from experiences and become more capable over time. Our contributions in this paper are (1) Identification of failure modes and reliability challenges in scalability, resilience, and learning using TritonBot. (2) Formulation of engineering practices that reduce manual interventions during long-term robot deployments. (3) Collection of design considerations that increase the reliability of long-term autonomous service robots. We tested the engineering practices and design considerations on TritonBot, but they are also applicable to other robot systems with scalability, resilience, and learning requirements.

The paper is organized as follows. Section 2 discusses related work about long-term autonomy and reliability engineering. Section 3 introduces the overall design and functions of TritonBot. Section 4 describes our efforts in making the TritonBot system scalable in its long-term evolvement (Scalability). Section 5 shows our practices in making TritonBot resilient to failures (Resilience). Section 6 presents our attempts to improve TritonBot over time both autonomously or with the help of developers (Learning). Finally, Sect. 7 concludes this paper.

2 Background and Related Work

A number of projects have studied long-term autonomy. The STRANDS project [12] deployed security patrol and guide robots and reached a few weeks of autonomy without intervention. CoBots [21] navigated over 1,000 km in open environments and intensively studied long-term mapping, localization, and navigation. BWIBots [14] is a custom-designed long-term multi-robot platform for AI, robotics, and human-robot interaction that aims to be a permanent fixture in a research facility. Early "roboceptionists" and tour guide robots like Valerie [15], RHINO [4], and Minerva [23] interacted and provided tours to visitors in schools and museums with an emphasis on social competence or robust navigation in crowded environments. Previous work also reported deployment experiences and failure statistics in other robot systems [6, 7]. These works provided valuable experiences of robots under challenging environments, but a few of them conclude lessons from a system-design perspective.

Reliability engineering is not a new concept in engineered systems; it is a critical study that keeps an engineered system reliable, available, maintainable, and safe [3]. Some companies in the industry even have created a *Site Reliability Engineer (SRE)* role in supporting growing Internet businesses [2]. Birolini [3] summarizes theories and provides qualitative approaches to the study of reliability, failure rate, maintainability, availability, safety, risk, quality, cost, liability, and so on. O'Connor et al. [18] further give field-specific reliability engineering examples of mechanical systems, electronic systems, software, design for reliability, manufacturing, and more. Beyer et al. [2] from Google Inc. discuss "site reliability engineering" and the principles and practices that keep Google's planet-scale production system healthy and scalable; they combine automated tools and appropriate engineering and emergency response workflows. None of these works are directly applicable to service robots given the gap between traditional computer systems and cyber-physical systems; yet despite the disparity between planet-scale datacenters and mobile robot platforms, successful engineering practices in traditional computer systems have considerably inspired robotic reliability engineering.

3 TritonBot Overview

Our goal in the TritonBot project is to identify reliability challenges in long-term autonomous service robots and incorporate appropriate reliability engineering methods from large-scale software and other engineered systems to confront these challenges. To this end, we built TritonBot as a realistic example of a long-term autonomous service robot: TritonBot has a number of advanced features including face recognition, voice recognition, natural language understanding, navigation, and so on; but it is not overly complicated, and thus it reflects the reliability issues that may occur in a commercial service robot in the near future. In a previous work [24], we presented the initial lessons learned from TritonBot in its initial deployment; we have been improving the TritonBot system to make it more robust since then.

TritonBot is based on the *Fetch Research Edition* platform from Fetch Robotics Inc. [26]. The robot has an RGBD camera with about 70° field of view, a microphone array with sound source localization support, and a loudspeaker. We installed a back-facing laser scanner at the opposite side of the stock laser scanner at the leg level; these laser scanners enable 360° leg detection and tracking. The base of the robot contains two differential drive wheels and a consumer-grade computer (Intel i5-4570S CPU,

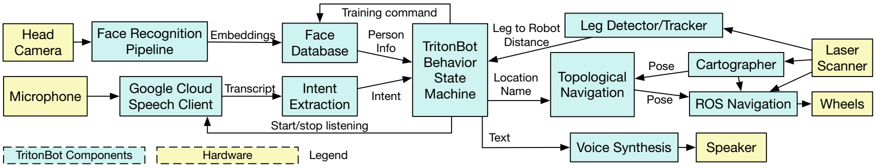

Head Camera
Face Recognition Pipeline
Embeddings
Face Database
Training command
Person Info
TritonBot Behavior State Machine
Leg to Robot Distance
Leg Detector/Tracker
Laser Scanner
Microphone
Google Cloud Speech Client
Transcript
Intent Extraction
Intent
Start/stop listening
Location Name
Topological Navigation
Pose
Cartographer
Pose
ROS Navigation
Wheels
Text
Voice Synthesis
Speaker
TritonBot Components
Hardware
Legend

Fig. 2 A block diagram shows the primary components in TritonBot and the data flow between them. A state machine controls the behavior of TritonBot, and standalone components including face recognition, voice recognition, leg detection and tracking, localization, and navigation support the TritonBot functions

16 GB memory, 1 TB solid-state disk); two lead-acid batteries keep the robot running for about 6 hours with a single charge.

TritonBot is deployed in Atkinson Hall, a research facility at UC San Diego. It stands in the hallway of the building, faces the entrance, and continuously detects faces in its view. When TritonBot detects a previously seen face, it greets the person by name. Otherwise, it briefly introduces itself, asks for the name of the visitor, and offers a trivia game. The simple questions allow the robot to interact with the visitor face-to-face and collect face samples. After the greetings and the games, TritonBot offers a guided tour to the user. The robot can show the visitor a few places of interest in the building, including a prototyping lab, a robot showroom, a "smart home" demo room, and a gallery. At each spot, the robot briefly introduces the place and confirms whether the visitor is still around using leg detection.

TritonBot uses the Robot Operating System (ROS), a standard middleware for robot prototypes for research. Figure 2 shows the software architecture in TritonBot. The system leverages the open-source software Cartographer [13] for localization and mapping, and the ROS navigation stack "movebase" [17] controls the robot to move around. A face recognition pipeline utilizes OpenFace [1] to generate face embeddings and calculate the similarity between two faces. An open-source leg tracker [16] detects and tracks human leg positions around the robot using laser scan ranges. TritonBot also leverages a cloud service Google Cloud Speech API [9] to convert a user's speech to a textual transcript; a commercial software package synthesizes TritonBot's voice. To summarize, we integrated many existing components together to build TritonBot, but wrote about 100,000 new lines of code (mostly C++ and Python) to build the entire TritonBot system.

4 Scaling up TritonBot over the Long-Term Deployment

Scalability is one of the three most important characters in long-term autonomy; it allows the developers to grow and expand a robot system without overhauling the existing architecture of the system and affecting normal operations. This section discusses our effort in making TritonBot scalable.

4.1 Scalability Challenges in TritonBot

TritonBot faced many challenges in scalability that add difficulties in its development and evolvement:

- Backward- and forward-compatibility: Scaling up requires fast software iteration. But the lack of backward- and forward-compatibility forces the developers to complete coding and testing on all related components before rolling out a new feature; rolling back one component also affects all related parts. Long-term logging also becomes a challenge when the developers add, update, or remove fields in the log format.
- Software architecture: Tightly coupled software limits the scalability of a computer system, but decoupling software components introduces difficulties in interfacing and coordination between the components. Besides, when TritonBot offloads computationally intensive software components to other machines, crossing network boundaries and communicating over the Internet bring in concerns in accessibility, latency, and confidentiality.
- Software management: With numerous robotic programs with different requirements running together to form a complete system, managing software running on the on-robot computer and other hardware is challenging. The limited computing power available on-board worsens the problem when computationally intensive programs compete for computing resources; they tend to create resource contention and exhaust all CPU or memory capabilities.

4.2 Forward- and Backward-Compatibility

Many robot systems use Robot Operating System (ROS) [20] to split the entire software system into multiple programs. ROS programs communicate through a publish-subscribe pattern to work with each other, and ROS provides an interface description language (ROS messages) to support the pub-sub framework. But any change to the message definition, no matter how insignificant it may be, invalidates all previously serialized data and generated libraries. Such inflexibility helps ROS to become a consistent community-maintained robotic software toolkit, but it limits the long-term evolution of an autonomous robot system, and it makes serializing ROS message a bad choice for long-term logging.

In TritonBot, we leverage an open-source libraries Protocol Buffer (ProtoBuf) [11] to provide forward- and backward-compatibility. ProtoBuf is a language-neutral, platform-neutral extensible mechanism for serializing structured data, but serialized ProtoBuf messages remain accessible even if the format description changes (in a compatible way). ProtoBuf can even work with RPC (remote procedure call) frameworks to provide backward compatibility between different programs. We use ProtoBuf to store sequential or small structured data, such as the topological map for navigation, face embedding of visitors, and long-term logs.

As an example, TritonBot leverages ProtoBuf to save its long-term log. We created a “Universal Logger” that stores a stream of ProtoBuf messages into compressed and size-capped files and distributes them into directories with a date and timestamp. Because ProtoBuf format is forward-compatible, adding extra fields to the log format does not affect previously stored logs; being backward-compatible, analysis programs written against an old ProtoBuf format will continue to work with newly generated logs. As an example, the voice recognition program on TritonBot initially only records voice recognition results that trigger the robot’s response; later, when we moved to a more capable voice recognition engine, we updated the data format to include the interim results. Thanks to the compatibility, the analysis script can still read previously stored logs. We collect about 1.2 GB ProtoBuf-based logs monthly on average, and we were able to generate many insights from the data; since the log format is optimized for machine-reading, scanning through all of the log entries only takes a few minutes.

4.3 Decoupling Software Components

In addition to the lack of backward- and forward-compatibility, ROS has two more limitations: the inflexible networking requirements, and the lack of security support. ROS programs (nodes) assume bidirectional direct TCP/UDP connection between each other, so they cannot communicate over networking environments with firewalls or network address translation (NAT) devices. In addition, ROS communication lacks encryption and authentication support, which prevent it from communicating over public channels such as the Internet. These three issues contribute to the software coupling challenges in scalability.

To overcome these shortcomings, we built a part of the TritonBot system with open-source libraries and gRPC [10] alongside ROS: gRPC is an open-source remote procedure call (RPC) framework that supports both synchronous or asynchronous, unary or streaming backward- and forward-compatible ProtoBuf messages in both requests and responses; it leverages HTTP/2 channels with optional encryption and authentication that can easily pass network devices. We built and released a ROS package `grpc` [22] that helps the users to generate, compile, and link ProtoBuf libraries and gRPC stubs within the ROS build environment.

The face recognition pipeline in the TritonBot system heavily leverages ProtoBuf and gRPC to maximize its scalability. Figure 3 shows the components and the data flow in the face recognition pipeline. Since the ProtoBuf format is forward-compatible, the face database file did not require any conversion when we added an option to store the face images along with the embeddings. ProtoBuf/gRPC format is also backward-compatible: when we updated the face database server, the old client continued to work, and we had a chance to test the new server before implementing a new client. In addition, gRPC helped us to offload the computationally intensive face embedding generation program from the robot: we put a face embedding generation server on a powerful host system behind a firewall (for network address translation)

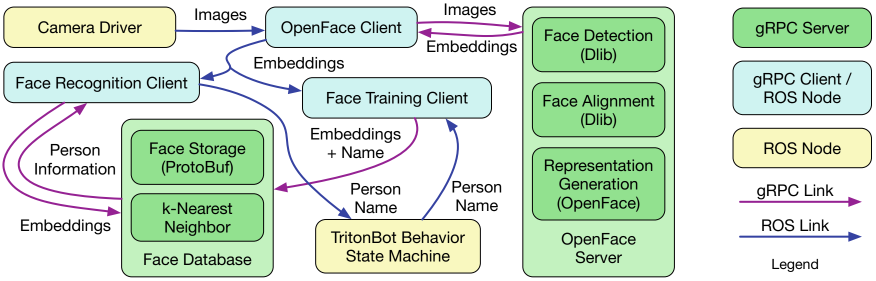

Camera Driver
Images
OpenFace Client
Images
Embeddings
Face Detection (Dlib)
Face Alignment (Dlib)
Representation Generation (OpenFace)
OpenFace Server
Face Recognition Client
Embeddings
Face Training Client
Embeddings + Name
Person Information
Embeddings
Face Storage (ProtoBuf)
k-Nearest Neighbor
Face Database
Person Name
Person Name
TritonBot Behavior State Machine
gRPC Server
gRPC Client / ROS Node
ROS Node
gRPC Link
ROS Link
Legend

Fig. 3 The face recognition pipeline in TritonBot. A few components in ROS bridge the ROS system with standalone gRPC servers, so that TritonBot exploits the existing open-source components in ROS and backward- and forward-compatibility from gRPC. After running TritonBot for some time, we moved the "OpenFace server" to a remote computer with powerful GPUs to save computing resources on the on-robot computer

and a reverse proxy (for authentication); the robot accessed the service over the Internet.

In general, design considerations in decoupling software speed up software iteration and increase the scalability of a robot system; backward- and forward-compatibility enable decoupling in many scenarios.

4.4 Managing Robotic Software with Linux Containers

In TritonBot, we use Linux containers to manage robotic software deployment and provide a unified development environment. The Linux container technology [19] provides a convenient approach to build, deploy, and run applications on different machines. Backed by the Linux namespaces that isolate and virtualize system resources for a set of processes, Linux containers are much lighter-weighted than fully virtualized kernels in hypervisors or virtual machines.

Initially, we adopted Docker [8], a popular and open-source container management tool to run every software component (41 in total) as separate containers that isolated execution environment for each of them. When TritonBot system grows larger, resource contention becomes an issue in scaling it up. In data centers, scaling up a software system means spawning more program instances and load-balancing the tasks among them; popular tools like Kubernetes [5] leverage Linux containers to provide a unified platform to manage the programs. A service robot like TritonBot only carries limited computing resources; however, it does not use all the components at the same time—for example, TritonBot does not face recognition results when it is moving, and vice versa. We built Rorg [25], an open-source container-based software management system. Rorg not only provides an accessible mechanism to use Linux containers in a robot programming environment, but it also models the component dependencies and decides what components to start to fulfill the robot's request. With the help of Rorg, the TritonBot uses 45% less CPU than before, which leaves the developer with plenty of room to add more features. Rorg also adds some additional benefits to service robots: because the software components have chances to stop and restart often, transient issues like memory leaks have a less significant outcome. This conclusion is consistent with the experience from Google that "a slowly crashlooping task is usually preferable to a task that has not been restarted at all" [2].

In conclusion, tools like Linux containers not only benefit traditional computer systems but also improve the scalability of service robots. The different use scenarios on service robots require the unique customization of the tools.

5 Tolerating and Coping with Failures

Resilience is the ability of a robot to adapt to environmental changes and tolerate or recover from transient errors. The TritonBot developers tried to make the robot as robust to failures as possible, and this section discusses our efforts.

5.1 Resilience Challenges in TritonBot

None of the engineered systems is immune from failures, and long-term deployments further expose the errors in a service robot. The TritonBot system has a few challenges to become resilient to failures.

- Transient failures: Some failures in service robots are transient and recoverable. For example, TritonBot sometimes loses its network connection when it enters and exits a WiFi blindspot, but a simple reconnecting attempt can effectively fix this issue. The two challenges in dealing with transient failures are (1) identifying the failures and (2) implementing the fix.
- Overloaded system: Almost all basic robot functionalities rely on the only programmable part of TritonBot, the on-robot computer. The computer affords too much functionality that a minor issue on the computer will lead to serious outcomes: when the Bluetooth stack on the computer fails, the developers lose the ability to drive the robot manually; if the computer encounters networking issues, the developers cannot connect to the computer.
- Lack of monitoring: Autonomous service robots are expected to work without close human supervision, but the developer does not have the means to understand the system characteristics and discover failures. However, too close supervision defeats the purpose of autonomy.

5.2 Recovering from Transient Failures

Frequent self-diagnosis and self-repair is a practical approach to discover and recover from some transient failures. TritonBot periodically runs some scripts to check and fix any potential issues. After TritonBot encountered WiFi issues multiple times, we created a script that checks the Internet connection (by pinging a commonly known website) and restarts the wireless interface in case of a failure. In another case, a Linux system service (`sssd`) sometimes gets killed and does not restart properly; we created another script to restart it in case of failure. However, the challenge in fixing transient failures is not making patches, but instead identifying the failure cases. Long-term deployments expose these transient failure patterns; occasional monitoring (Sect. 5.4) helps the developers to capture these failures.

While hand-made scripts are useful in complementary to system service management tools, another type of transient failure that happens in robotic software is that some programs get killed when they encounter unhandled exceptions or have design flaws. In TritonBot, any unexpected program crash triggers restarting itself or its programs group. While crashlooping programs often indicate issues in the system, following proper design principles, we observed that some infrequent program restarts have little effect on the overall reliability: in the early TritonBot development, we had an issue that the voice recognition software crashes and restarts every few hours. The system continued to work (possibly with a few unnoticed attempts to retry voice recognition); only later did we find the unusual restarts in the logs and fixed a memory leaking issue. We have concluded three design principles to handle transient failures in robotic software: (1) Never to hide failures that fail silently; it is better to crash a program and expose issues. (2) Restarting a program should help it to enter a clean state and recover from transient failures. (3) Any software component should tolerate the unavailability of another component.

5.3 Relying on Separate Subsystems

Figure 4 shows the hardware components in TritonBot. The bare Fetch Research Edition platform only has two parts in its system architecture: an embedded computer to run user programs, and a low-level controller to control the actuators. When we built

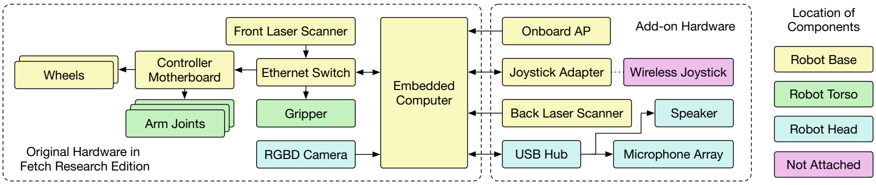

Front Laser Scanner
Wheels
Controller Motherboard
Ethernet Switch
Embedded Computer
Arm Joints
Gripper
RGBD Camera
Original Hardware in Fetch Research Edition
Onboard AP
Add-on Hardware
Joystick Adapter
Wireless Joystick
Back Laser Scanner
Speaker
USB Hub
Microphone Array
Location of Components
Robot Base
Robot Torso
Robot Head
Not Attached

Fig. 4 Main hardware components in the TritonBot system. We added some additional hardware to the Fetch Research Edition

the early TritonBot prototype, we relied solely on programming and configuring the embedded computer. Soon, we found that the on-robot computer became a frequent point of failure. In the TritonBot evolution, we added a wireless access point to the robot as an “escape hatch” in addition to the regular encrypted access channel over the Internet. We also paired the manual-override gamepad directly with a USB dongle that emulates a joystick device instead of the Bluetooth radio in the embedded computer. As a general design principle, we found that relying on dedicated systems reduces single-point failures and prepares the robot and the developer for unexpected situations.

5.4 Monitoring the System

The challenge in fixing transient failures it not making patches, but instead identifying the failure cases; occasional monitoring of TritonBot during a long-term deployment helps us to achieve such a goal. In the TritonBot project, we mainly use two monitoring tools: First, we built an Android app to see the robot’s view and battery status. Second, the robot analyzes its log and sends a summary to the developers every night.

The robot monitor (Fig. 5a) is an Android tablet. The robot captures its battery level, charging status, and camera images every few seconds, and it sends them to a central server through an authenticated channel over the Internet. The tablet displays these states of the robot and gives the developer an overview of the robot status with a simple glance. In addition to the monitoring tablet, TritonBot also analyzes its daily log and sends an E-mail about its daily work summary to the developers every mid-night. The report includes the interaction transcript and a summary of its daily

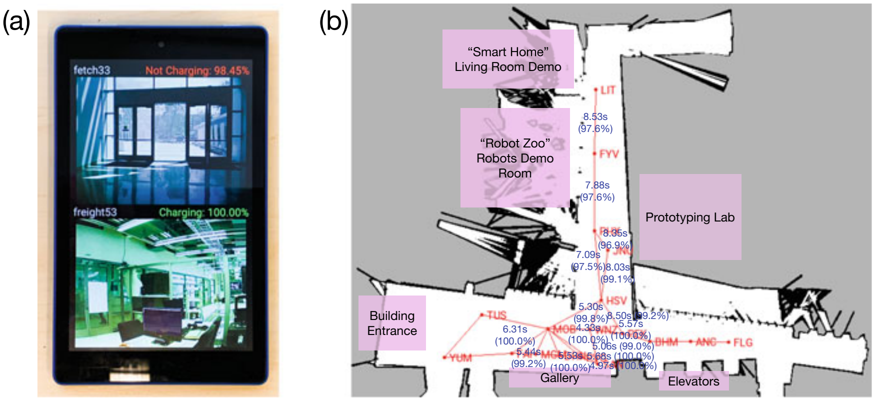

(a)
fetch33
Not Charging: 98.45%
freight53
Charging: 100.00%
(b)
“Smart Home” Living Room Demo
“Robot Zoo” Robots Demo Room
Prototyping Lab
Building Entrance
Gallery
Elevators
LIT
8.53s (97.6%)
FYV
7.88s (97.6%)
8.35s (96.9%)
JNU
7.09s (97.5%)
8.03s (99.1%)
HSV
5.30s (99.8%)
8.50s (99.2%)
5.57s (100.0%)
TUS
6.31s (100.0%)
MOB
4.33s (100.0%)
5.06s (99.0%)
BHM
ANC
FLG
YUM
5.44s (99.2%)
5.53s (100.0%)
5.68s (100.0%)
4.97s (100.0%)

Fig. 5 **a** An Android tablet that displays the robot’s eye view and battery status. The developer occasionally monitors the general status of the robot during the deployment. **b** The navigation topological map of TritonBot. The map shows the average time to traverse an “airway” on the map as well as the success rate events such as the number of humans engaged, trip traveled, and so on. Reading the E-mail allows the developers to understand the robot's performance in general, while more detailed logs generated by different components are also available for further analysis. Both of the monitoring methods retain the robot's autonomy, but they allow the developers and the users to understand the robot's status at different levels.

6 Learning from the Past

Learning in long-term autonomy suggests that a system can learn from the past and improve its performance. TritonBot learns from the deployment experience and improves itself over time. In addition to the robot learning by itself, the TritonBot developers also learn from past failures: we created a fault injection tool to recreate past failure scenes on TritonBot, study rare failure conditions, and improve the robot system.

6.1 Learning Challenges in TritonBot

TritonBot has two primary learning-related issues:

- Repeated failures: TritonBot moves around when it shows the visitor the places of interest in the building. However, we have observed that it gets stuck on some paths at a significantly higher rate than others. Even with much experience of similar failures, it continues to make the same mistakes because it always plans for the shortest path.
- Rare failure types: Some failures on TritonBot are triggered with a few "coincidences." These failures are difficult to reproduce, which prevents the developers from making an in-depth investigation of the failures; when the developers come up with potential fixes, there is no practical method to verify the fixes.

6.2 Learning from Long-Term Deployment

Unlike short-term demos, long-term autonomous service robots accumulate experience over time. These experiences can turn into precious knowledge that improves the robustness of the robot. TritonBot applies this idea to navigation—when it moves around, it records the time to travel each path on a map, and avoids the paths that took too long to travel when generating a move plan.

TritonBot moves around to show the places of interest to the visitors. On top of the classic *movebase* navigation stack from ROS [17], we added a topological map layer of waypoints and paths (Fig. 5b). The core of learning is a "traffic control" service: when TritonBot decides to move to a waypoint, it requests a "traffic control" service to generate a plan—a list of waypoints connected by paths. TritonBot calls *movebase* to execute the plan with some preset time limit and error allowance. After traveling each path, TritonBot reports the travel time back to the traffic control service. The traffic control service internally adjusts the cost of each path according to the feedback, and the change affects the future plans. Figure 5b also presents the average traversal time and success rate of each path in a previous TritonBot deployment.

In the TritonBot deployment, using past experience to improve the system is a fundamental design principle. In another example, TritonBot records all of the utterances that it cannot understand during its conversations with people, and the developers use these utterances to improve its intent extraction algorithm. In conclusion, learning from past failures is a convenient and effective way to increase reliability over time in long-term robot deployments.

6.3 Learning from Rare Failures

Software fault injection is a conventional technique in software engineering; it intentionally introduces failure to test a system's response to failures. We created a fault injection tool "RoboVac" to inject failures to the TritonBot system. It helps us to find unknown design flaws, verify fixes, and benchmark the system's resilience.

RoboVac offers a unified framework for general fault injection needs on service robots. It leverages the ROS message passing framework (topics) to simulate failures in sensors, actuators, or even between software; it also enables the developer to inject failures at the general Linux process level to simulate a program crashing or stuck; it can reshape the network traffic to simulate different networking conditions. RoboVac offers an efficient workflow for the developers to improve a robot's performance under failures. With RoboVac, we were able to inject failures in the software, ROS message passing, networking, and ad hoc components. During the TritonBot evolution, we found many unseen design flaws using fault injection, and we used RoboVac to verify our fixes. We are continuing to work on RoboVac to offer a fuzz-testing scheme that enables automatic error discovery on service robots.

7 Conclusion

This paper presents robotic reliability engineering under the long-term autonomy scenario that a service robot works and evolves for an extended period. We discuss our reliability engineering practices in the context of TritonBot, a tour guide and receptionist robot in a university building. As a long-term autonomous service robot, it has challenges in the three aspects of long-term autonomy: scalability, resilience, and learning. TritonBot optimizes data compatibility, software architecture, and resource management to retain scalability. It tolerates transient failures, avoids single-point failures, and leverages monitoring to improve its resilience. It also learns from the experience and takes advantage of software fault injection. All these efforts increase the reliability of TritonBot.

Failures in service robots are unavoidable but manageable. TritonBot provides a realistic example of a long-term autonomous service robot in an open environment. We will continue to deploy TritonBot to provide more experiences and insights into running a long-term autonomous service robot. The complete TritonBot source code is available at https://github.com/CogRob/TritonBot. We hope that the reliability engineering techniques and experiences from TritonBot will inspire more robust long-term autonomous service robot designs.

References

1. Amos, B., Ludwiczuk, B., Satyanarayanan, M.: Openface: a general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118, CMU School of Computer Science (2016)
2. Beyer, B., Jones, C., Petoff, J., Murphy, N.: Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, Incorporated (2016)
3. Birolini, A.: Reliability Engineering, vol. 5. Springer (2007)
4. Burgard, W., Cremers, A.B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: The interactive museum tour-guide robot. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI '98/IAAI '98, pp. 11–18. American Association for Artificial Intelligence, Menlo Park, CA, USA (1998)
5. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., Wilkes, J.: Borg, omega, and kubernetes. Commun. ACM **59**(5), 50–57 (2016). https://doi.org/10.1145/2890784
6. Carlson, J., Murphy, R.R.: Reliability analysis of mobile robots. In: 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), vol. 1, pp. 274–281 (2003). https://doi.org/10.1109/ROBOT.2003.1241608
7. Chung, M.J.Y., Huang, J., Takayama, L., Lau, T., Cakmak, M.: Iterative design of a system for programming socially interactive service robots. In: Agah, A., Cabibihan, J.J., Howard, A.M., Salichs, M.A., He, H. (eds.) Social Robotics, pp. 919–929. Springer International Publishing (2016)
8. Docker Inc.: Docker is an open platform to build, ship and run distributed applications anywhere (2018). https://www.docker.com
9. Google LLC: Cloud speech API—speech to text conversion powered by machine learning (2017). URL https://cloud.google.com/speech
10. Google LLC: gRPC: A high performance, open-source universal RPC framework (2018). https://grpc.io/
11. Google LLC: Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data (2018). https://developers.google.com/protocol-buffers/
12. Hawes, N., Burbridge, C., Jovan, F., Kunze, L., Lacerda, B., Mudrova, L., Young, J., Wyatt, J., Hebesberger, D., Kortner, T., Ambrus, R., Bore, N., Folkesson, J., Jensfelt, P., Beyer, L., Hermans, A., Leibe, B., Aldoma, A., Faulhammer, T., Zillich, M., Vincze, M., Chinellato, E., Al-Omari, M., Duckworth, P., Gatsoulis, Y., Hogg, D.C., Cohn, A.G., Dondrup, C., Fentanes, J.P., Krajnik, T., Santos, J.M., Duckett, T., Hanheide, M.: The strands project: long-term autonomy in everyday environments. IEEE Robot. Autom. Mag. **24**(3), 146–156 (2017). https://doi.org/10.1109/MRA.2016.2636359

13. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2d lidar slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1271–1278 (2016). https://doi.org/10.1109/ICRA.2016.7487258
14. Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V., K. Aggarwal, J., Mooney, R., Stone, P.: Bwibots: A platform for bridging the gap between ai and human-robot interaction research. Int. J. Robot. Res. **36**, 635–659 (2017)
15. Kirby, R., Forlizzi, J., Simmons, R.: Affective social robots. In: Robotics and Autonomous Systems, vol. 58. Pittsburgh, PA (2010)
16. Leigh, A., Pineau, J., Olmedo, N., Zhang, H.: Person tracking and following with 2d laser scanners. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 726–733 (2015). https://doi.org/10.1109/ICRA.2015.7139259
17. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: robust navigation in an indoor office environment. In: 2010 IEEE International Conference on Robotics and Automation, pp. 300–307 (2010). https://doi.org/10.1109/ROBOT.2010.5509725
18. O'Connor, P., Kleyner, A.: Practical Reliability Engineering. Wiley (2012)
19. Pahl, C., Lee, B.: Containers and clusters for edge cloud architectures—a technology review. In: 2015 3rd International Conference on Future Internet of Things and Cloud, pp. 379–386. IEEE (2015)
20. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5. Kobe (2009)
21. Rosenthal, S., Veloso, M.M.: Mixed-initiative long-term interactions with an all-day-companion robot. In: AAAI Fall Symposium: Dialog with Robots, vol. 10, p. 05 (2010)
22. The Regents of the University of California: gRPC: Catkinized GRPC package (2019). https://github.com/CogRob/catkin_grpc
23. Thrun, S., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: Minerva: a second-generation museum tour-guide robot. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), vol. 3, pp. 1999–2005 (1999). https://doi.org/10.1109/ROBOT.1999.770401
24. Wang, S., Christensen, H.I.: Tritonbot: First lessons learned from deployment of a long-term autonomy tour guide robot. In: Proceedings of the 2018 IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 158–165 (2018)
25. Wang, S., Liu, X., Zhao, J., Christensen, H.I.: Rorg: Service robot software management with linux containers. In: 2019 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2019)
26. Wise, M., Ferguson, M., King, D., Diehr, E., Dymesich, D.: Fetch & freight: standard platforms for service robot applications. In: Workshop on Autonomous Mobile Service Robots, International Joint Conference on Artificial Intelligence (2016)

Autonomous Driving of Six-Wheeled Dump Truck with a Retrofitted Robot

Tomohiro Komatsu, Yota Konno, Seiga Kiribayashi, Keiji Nagatani, Takahiro Suzuki, Kazunori Ohno, Taro Suzuki, Naoto Miyamoto, Yukinori Shibata, and Kimitaka Asano

1 Introduction

Currently, the declining birthrate and aging population have become social problems in Japan . In the construction industry, this problem is serious and labor shortage due to aging will occur. Therefore, expectations for autonomous construction machines are increasing. Currently, in Japan, promoting the automation of construction machinery is a national policy. Ministry of Land, Infrastructure, Transport and Tourism (MLIT) has established policies such as information construction and i-Construction [1]. According to the above background, major construction machine manufacturers and major construction companies are researching and developing the automation of con-

---

T. Komatsu (✉)
Kowatech Co., Ltd., 5 -18 -18, Samukawa -Ichinomiya, Kozagun, Kanagawa, Japan
e-mail: komatsu@frl.mech.tohoku.ac.jp

Y. Konno · S. Kiribayashi · T. Suzuki
Tohoku University, Sendai, Japan

K. Ohno
RIKEN AIP, Tohoku University, Sendai, Japan

N. Miyamoto
Tohoku University, 6-6-10, Aramaki-Aoba, Sendai, Japan

K. Nagatani
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan

T. Suzuki
Chiba Institute of Technology, 2-17-1 Tsudanuma, Narashino, Chiba, Japan

Y. Shibata
Sato Koumuten Co., Ltd., 69, Kami-Nagadan, Kamigun, Miyagi, Japan

K. Asano
Sanyo-Technics Co., Ltd., 3-1-40, Nigatake, Sendai, Miyagi, Japan

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_5

struction machinery [2, 3]. However, in local companies, only a few cases exist where autonomous construction machines, or smart construction machines, are introduced to regional construction companies.

Therefore, the authors aim to realize automated construction machines owned by regional construction companies. Particularly, in this research, the target is a 30-ton six-wheeled dump truck. The task to realize is earth and sand transportation from a certain loading position to an unloading position. This article reports an examination of functions required for the autonomous driving of the existing six-wheeled dump trucks, development of control modules, and verification using an actual truck with the modules. It includes the following four topics:

1. development of a robotic control module for conventional dump trucks,
2. development of in-vehicle sensor modules according to vehicle characteristics,
3. construction of control system to integrate the above and derivation of kinematics for a six-wheeled dump truck, and
4. consideration of a method to detect obstacles with three-dimensional (3D) Light Detection and Ranging (LiDAR).

The effectiveness of the above was confirmed by initial experiments on an actual 30-ton dump truck.

2 Robotization of Existing Six-Wheeled Dump Truck

This research aims to realize autonomous driving for conventional six-wheeled dump trucks owned by regional construction companies. The target task for the dump trucks is earth and sand transportation from a certain loading position to an unloading position. Functions to realize autonomous driving are classified into the following four categories:

- Vehicle operation to handle the steering wheel, accelerator, and brake;
- Acquisition of vehicle information for vehicle control such as speed and position of a dump truck;
- Acquisition of environmental information to recognize surrounding environments such as obstacles and driving road surface;
- Control system to integrate the above functions.

The authors' approaches to realize the above functions are introduced in the following subsections.

2.1 Development of Retrofitted Robot: SAM

To realize the functions for the vehicle operation of dump trucks, it is necessary to handle the steering wheel, accelerator, and brake pedal. Some methods can be used to operate a vehicle directly by controlling its hydraulic system or electric signals such

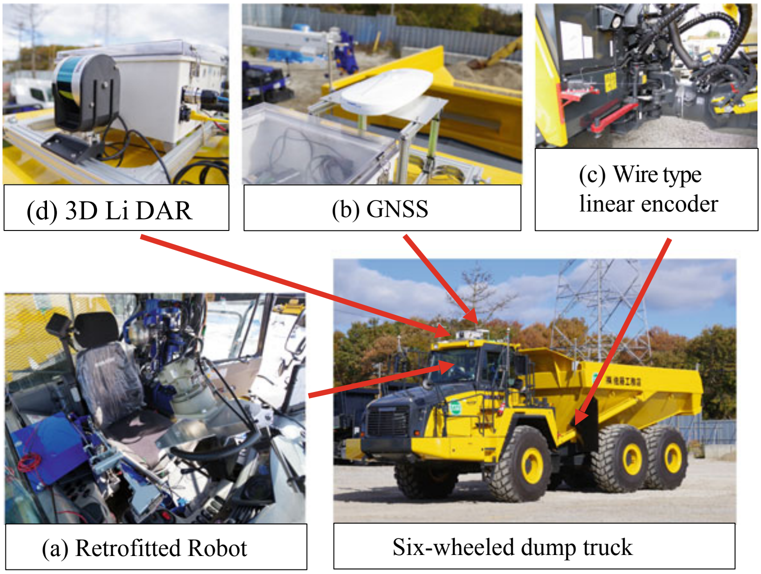


Fig. 1 Six-wheeled dump truck and equipment for autonomous driving

as CAN communication [4, 5]. However, it is difficult because vehicle information is not disclosed by construction companies, typically.

Therefore, in this research, a robot is retrofitted to the vehicle to physically drive the dump truck. Consequently, a significant modification of the vehicle is not required, and the robot can be developed by physical vehicle information. In this paper, the development of a robot on the driver's sheet, particularly to enable the operations of steering, accelerator, and brake pedal, is focused on the following.

Figure 1a shows the robot, called SAM. SAM is commercialized as a retrofitted system that enables the remote control of an existing hydraulic excavator [6, 7]. In this research, we reconfigured a commercialized SAM to control a dump truck (Fig. 2). SAM includes a pneumatic servo system that exhibits mechanical compliance from the compressibility of air such that it is considered effective against disturbances such as vibration and shock to construction machines. In field experiments, as shown in Sect. 4, the main body of SAM was installed on the passenger seat such that the person in the driver's seat can operate the brake for an emergency stop.

2.1.1 Operation Unit for the Steering Wheel

In the steering of the six-wheeled dump truck, the articulated angle changes according to the amount of rotation of the steering wheel. However, the neutral position of the steering is not constant. This is because the control amount of the steering wheel is that of the hydraulic pressure that controls the articulation mechanism, and not

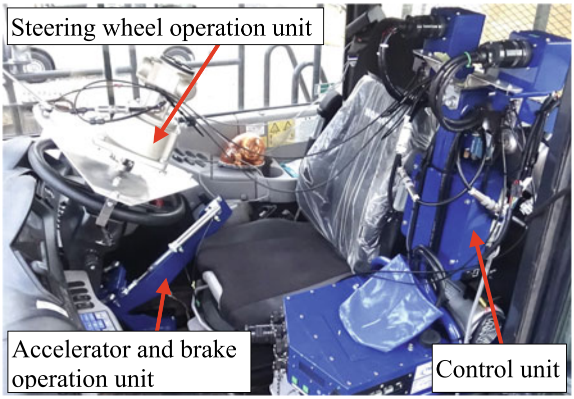

Steering wheel operation unit
Accelerator and brake operation unit
Control unit

Fig. 2 Retrofitted robot, SAM, which is located in passenger's seat

the articulation angle itself. In addition, the handle is structured to rotate infinitely. Therefore, a radial piston air motor with no limit in rotation range was adopted for the steering wheel operation. The air motor is operated by compressed air; therefore, it is resistant to overloads and external impacts during steering. In the steering drive mechanism, an air motor installed at the top of the steering wheel rotates the steering shaft directly. The rotational speed can be controlled by changing the airflow rate of the air motor.

2.1.2 Operation Unit for Accelerator and Brake Pedals

The vehicle speed of a six-wheeled dump truck is controlled by the amount of depression of the accelerator and brake pedals. Therefore, a pneumatic servo system was built on the pneumatic cylinder for the operation of the accelerator and brake pedal. For safety reasons, when all the power in the system is lost, the air in the cylinder is exhausted, and the spring-push-type pneumatic cylinder is configured to depress the brake pedal automatically.

2.2 Sensing Unit for Acquisition of Vehicle Status

In recent dump trucks, their speed and steering angle are measured in the system. However, practically, it is difficult to obtain information for researchers/operators because it is not disclosed. Therefore, the authors developed a sensor unit that obtains the position, heading, and articulated angle of the vehicle, and can be installed easily on the dump truck.

Figure 1b shows the GNSS unit (manufactured by Hemisphere: ssV-102) [8]. The system is equipped with two GNSSs such that it acquires the position and heading of the dump truck. Further, the velocity of the vehicle is calculated from the difference between the measured positions and times. The GNSS was located at the center of the front wheel shaft of the vehicle.

To acquire the articulate angle, which is equivalent to the steering angle, a wire-type linear encoder (manufactured by MUTOH: DE-04-C) [9] was used, as shown in Fig. 1c. Figure 3 shows a schematic diagram of the measurement method of the bending angle. From the measurement results, it was confirmed that there was linearity between the wire length obtained by the linear encoder and the articulated angle. However, since the wire-type sensor is not reliable enough for operation at construction sites such as rocky or dirt environments, it is assumed to be a tentative method. Another better method, e.g., using a laser range sensor, will be replaced in the future.

2.3 Path Tracking Method for Six-Wheeled Dump Truck

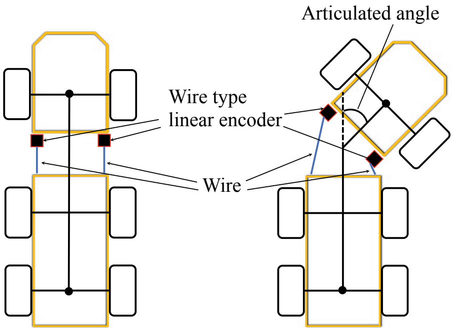
To realize the autonomous driving of a dump truck that does not collide with obstacles, path planning is important. In this study, it is assumed that the path is planned, and mainly the method of path tracking control is focused. The path tracking method, which the authors chose in this project, is the Pure Pursuit method [10, 11] applied to autonomous vehicles for an easy gain tuning.

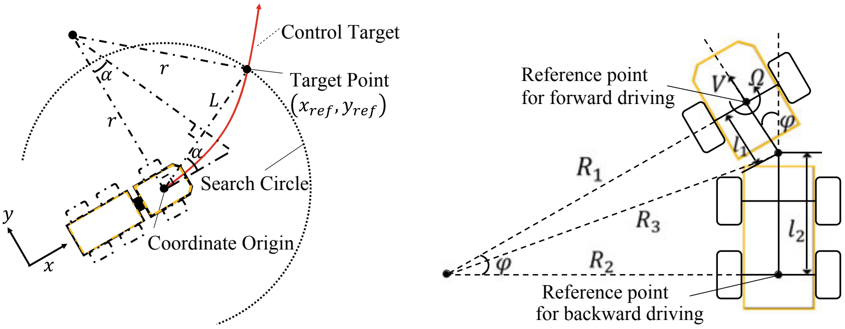
Figure 4a shows the conceptual diagram of the pure pursuit path tracking algorithm. The target point is determined from the current vehicle velocity V and a gain parameter d. The radius of the circle to search for the target point from the current position is expressed as follows:

$$L = Vd. \tag{1}$$

The angle α between a straight line passing through the target point (x_{ref}, y_{ref}) and the current position (= coordinate origin) and the heading line of the vehicle is expressed as follows:

Fig. 3 Schematic diagram of the measurement method of bending angle




Control Target
Target Point
(x_{ref}, y_{ref})
α
r
L
r
α
Search Circle
y
x
Coordinate Origin
Reference point
for forward driving
V
Ω
φ
R_1
l_1
R_3
l_2
φ
R_2
Reference point
for backward driving

(a) Conceptual diagram of pure pursuit method

(b) Flexion of six-wheeled dump truck.

Fig. 4 Path planning method

$$\alpha = \tan^{-1}\left(\frac{y_{ref}}{x_{ref}}\right). \tag{2}$$

The reference path is generated as a tangential line to the current vehicle heading and passes through both the target point and current position. Subsequently, the rotation radius of the vehicle r is expressed as the following equation from geometric relationships:

$$r = \frac{L}{2\sin\alpha}. \tag{3}$$

Therefore, the reference rotational velocity of the vehicle Ω_{ref} is expressed as follows:

$$\Omega_{ref} = \frac{V}{r} = \frac{2V\sin\alpha}{L}. \tag{4}$$

2.3.1 Kinematics of Six-Wheeled Dump Truck

The kinematics of the six-wheeled dump truck was derived to calculate the rotational velocity, which is the operation input for a vehicle, from the articulated angle of the vehicle. Figure 4b shows the kinematics of the six-wheeled dump truck. The geometrical relationship of the vehicle is expressed as follows:

$$R_1^2 + l_1^2 = R_2^2 + l_2^2 = R_3^2, \tag{5}$$

$$R_2 = R_1\cos\varphi + l_1\sin\varphi, \tag{6}$$

where R_1 denotes the length between the center of the front axle and the rotational center; R_2 is the length between the center of the rear axle and the rotational center; R_3 is the length between the hinge point and rotational center; l_1 is the length between the front axle and hinge point; l_2 is the length between the rear axle and hinge point; and φ the articulated angle.

According to Eqs. 5 and 6, R_1 is expressed as follows:

$$R_1 = \frac{l_1 \cos\varphi + l_2}{\sin\varphi}. \tag{7}$$

Then, the rotational velocity Ω_{now} is expressed as follows:

$$\Omega_{now} = \frac{V}{R_1} = \frac{V \sin\varphi}{l_1 \cos\varphi + l_2}. \tag{8}$$

Thus, the relationship between the rotational velocity and the articulated angle of the six-wheeled dump truck is derived. Therefore, the values for vehicle control are obtained by the kinematics.

2.3.2 Forward and Backward Path Tracking

In the target task, for transporting soil from the loading position to the unloading position, both forward and backward driving are required for a dump truck. In forward driving, the center of the front wheel shaft is defined as a reference point. However, in backward driving, the center of the rear wheel shaft of the vehicle is regarded as the reference point. This reference point is regarded as the current position of the vehicle in the Pure Pursuit method. The position and heading of the rear wheel center are derived using the position and heading of the GNSS, articulate joint angle, length l_1, and length l_2.

3 Obstacle Detection

3.1 Acquisition Function of Environmental Information Based on 3D LiDAR

For the safe driving of an autonomous dump truck, it is necessary to confirm the condition of the road surface and the existence of obstacles. Acquiring a 3D topography map and updating it is effective to enable the above. Therefore, a 3D LiDAR is typically used to acquire environmental information [12, 13]. In this research, a safety function based on the 3D LiDAR is introduced to confirm the safety in front of the dump truck where human entry cannot be completely restricted in a relatively common construction site.

First, a suitable installation configuration of the 3D LiDAR is examined for the six-wheeled dump truck and the driving environment in construction sites. Next, an algorithm for obstacle detection with the LiDAR was implemented. In the initial examination, verification of the forward obstacle detection was carried out, and rear obstacle detection is in future works.

3.2 Examination of Suitable Installation Configuration of 3D LiDAR

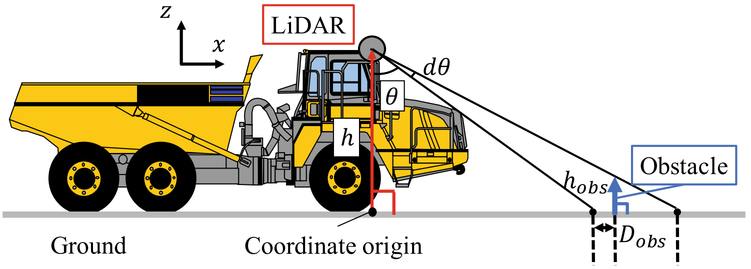
In this research, the device to obtain environmental information is the 3D LiDAR, manufactured by Velodyne Lidar (VLP-16) [14]. Figure 1d shows the LiDAR installed on the vehicle. Obstacles are detected as point cloud data acquired from the 3D LiDAR. In the measurement range of the 3D LiDAR, 16 lasers are emitted at intervals of 2° in the vertical direction, and the measurement range is 30°. In the horizontal direction, the laser is emitted at an interval of 0.2° in all directions. The LiDAR was installed on the roof because the front can be sufficiently overlooked without being affected by the road surface irregularities and it can be easily detached. Typically, the LiDAR (VLP-16) is installed horizontally to obtain the environmental information of all surroundings. However, in this research, the installation direction of the LiDAR was vertical to increase the resolution of the traveling direction.

3.3 Derivation of Unmeasurable Region

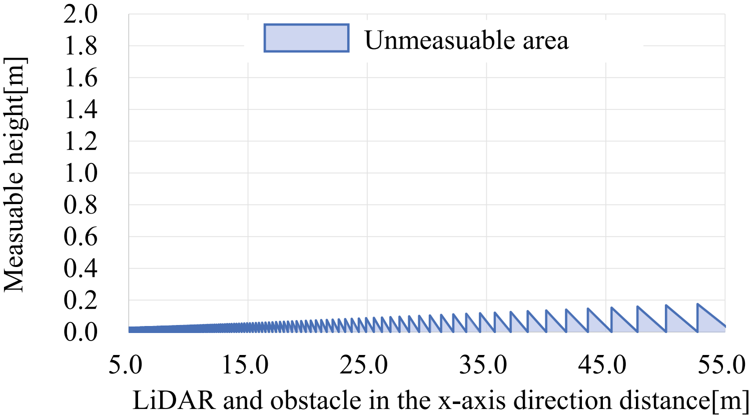
To confirm the limitation of the measurement region when the LiDAR is placed vertically, the unmeasurable region is calculated geometrically from the positional relationship between the LiDAR and ground. Figure 5a shows the measurement of obstacles from the road surface upwards. Here, h m denotes the installation height of the LiDAR from the ground; h_{obs} m is the height of the measurable obstacle; $\theta°$ is the angle between the perpendicular drawn from the LiDAR to the ground and the straight line of the emitted laser; $d\theta°$ is the angular resolution of the LiDAR; and D_{obs} m is the distance between the point at which the emitted laser crosses the ground and the obstacle in the x-axis direction. From the triangle approximation, the height of the detectable obstacle h_{obs} is expressed as follows:

$$h : h \tan(\theta + d\theta) = h_{obs} : h \tan(\theta + d\theta) - h \tan\theta - D_{obs} \tag{9}$$

$$h_{obs} = \frac{h \tan(\theta + d\theta) - h \tan\theta - D_{obs}}{\tan(\theta + d\theta)}. \tag{10}$$



(a) Diagram of positional relationship between 3D LiDAR installed in dump truck and obstacle.



(b) Graph of unmeasurable region.

Fig. 5 Diagram of LiDAR installation position and unmeasured area

The parameters in this verification test are $h = 3.5$ m, $d\theta = 0.2°$. Figure 5b shows an unmeasurable region calculated from the measurable height. The unmeasurable height is approximately 0.2 m at 50 m.

3.4 Obstacle Detection Method

In this research, for simple and robust obstacle detection, an obstacle is assumed if the gap dz in the z-axis direction between adjacent point cloud data is larger than the threshold dz_{th}. The calculation is performed independently for each of the 16 lasers to be emitted. In the lateral direction, each laser gap is the minimum recognition width. For example, at the point of 10 m from the laser range finder, the minimum recognition width is 0.35 m. Because this method involves a simple comparison of height between two points, it is effective for long objects such as people and signboards targeted in this verification.

4 Initial Field Experiments

To confirm the effectiveness of the proposed method, two initial field experiments were conducted using the developed device and an articulated dump truck.

Both experiments were conducted in Sato Koumuten's field in Kami-gun, Miyagi prefecture. The environment was a flat surface. The target vehicle in both experiments was HM300-5 (manufactured by Komatsu) [15]. The details are introduced in the following subsections.

4.1 Experiments of Forward and Backward Path Tracking

4.1.1 Purpose of Experiment and Conditions

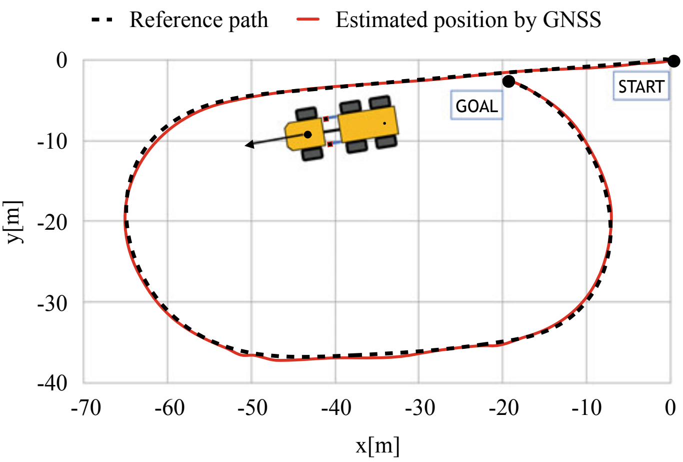
In this experiment, the performance of the path tracking system shown in Sect. 2 is evaluated by comparing the error between the reference path and the executed path. The reference path of the dump truck, which is a list of GNSS coordinates, was generated by manually driving the vehicle in advance. The translational reference velocity was 6 km/h for forward path tracking and 4 km/h for backward path tracking. The distance from the current position obtained by GNSS to the closest point on the reference path is defined as the tracking error. In this experiment, the gain parameter d was determined by trial and error.

4.1.2 Experimental Results and Discussions

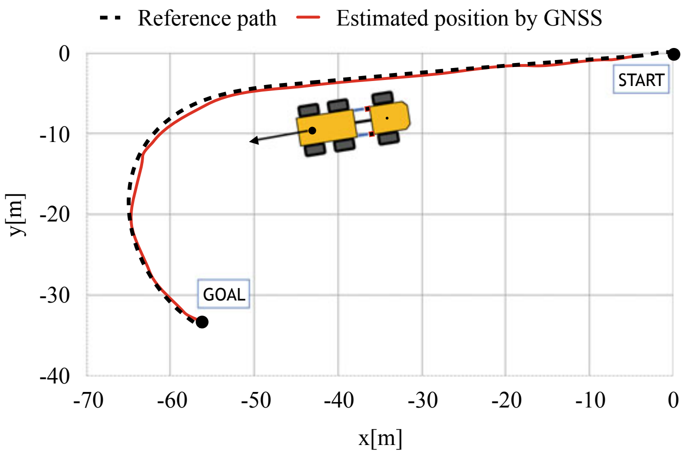
Figure 6a and b shows the experimental results of forward and backward path tracking. The starting point is the coordinate origin (0, 0). The dotted line is the reference path, and the solid line is the path trajectory estimated by the GNSS.

Based on Fig. 6a, the maximum error between the reference path and the actual path was 0.5 m for forward path tracking. However, based on Fig. 6b, the maximum error between the reference path and the actual path was 1.0 m for backward path tracking.

The magnitude of the error is considered to be within the acceptable range. However, the error between the target path and the actual trajectory for backward path tracking was larger than that for forward path tracking. The reason is as follows. In backward path tracking, the (x, y) position of the origin of the dump truck is located in the middle of the rear wheels. The position is calculated by the position of the GNSS installed on the front roof of the dump truck and the bending angle of the articulate mechanism. The measurement of the articulate angle is now performed by wire linear encoders, but includes errors. Therefore, the angular error affects the positioning error of the origin of the dump truck in backward path tracking.



(a) Experiment results of forward path tracking.



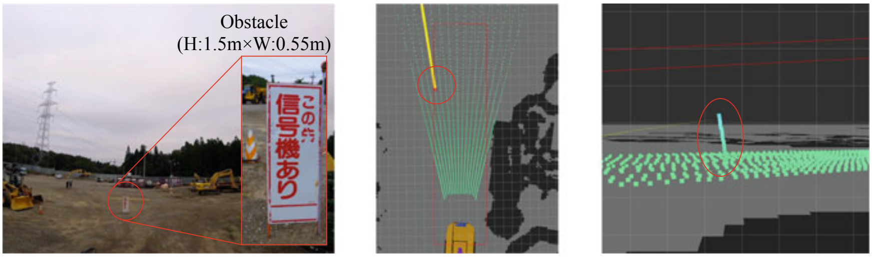
(b) Experiment results of backward path tracking.

Fig. 6 Results of autonomous path tracking experiment

4.2 Emergency Stop Experiment by Detection of Obstacles

4.2.1 Purpose of Experiment and Conditions

To realize the safe driving of the six-wheeled dump truck, an obstacle detection, as shown in Sect. 3.4, and an emergency stop function were implemented. The purpose of the experiment is to confirm whether the simple proposed method operates correctly in a real field with a real dump truck.



(a) Camera image.

(b) Measurement screen.

(c) Measurement screen enlarged with changing viewpoint.

Fig. 7 Obstacle detection result in case of emergency stop

The LiDAR was installed at the top of the roof of the driver's seat. The height was 3.6 m from the ground. The target obstacle was a signboard of a construction site with a height of 1.5 m and width of 0.55 m. The translational reference speed of the dump truck was 6 km/h. The threshold dz_{th} was set to 0.7 m. In this experiment, a dump truck executed a forward path tracking motion on a road where obstacles were placed in advance. The detection region of the obstacles was set as 0 to 25 m in the depth direction and -3 to 3 m along the width of the road, in the horizontal direction from the LiDAR. When an obstacle is detected within this range, the vehicle stops autonomously.

4.2.2 Experimental Results

Before the emergency stop experiments, a static capability verification of the 3D LiDAR was conducted. The target obstacle was set 50 m away from the dump truck, and the detection function replied detection of it correctly. Therefore, in consideration of the margin, it was concluded that the system was acceptable for the detection of obstacles within 25 m.

Figure 7 shows the experimental results. The results of the emergency stop experiments are introduced in the following. Figure 7a shows the image obtained by the camera installed on the roof of the dump truck, Fig. 7b shows the point cloud data drawn from the top of the vehicle, and Fig. 7c shows the point cloud data drawn from the side of the vehicle. The mesh width of Fig. 7b and c is 1 m. In Fig. 7b, the dotted green line shows the scanning line of the LiDAR, the red frame shows the obstacle detection range, the red circle shows the point where the obstacle is detected, and the solid yellow line shows the scanning line blocked by the obstacle. According to Fig. 7b and c, changes due to the obstacles were confirmed in the point cloud data obtained from the LiDAR. Subsequently, the dump truck stopped autonomously as soon as the detected part was located in the obstacle detection range. In addition, it was confirmed that the dump truck also stopped by the detection of a human as an obstacle.

Consequently, it was confirmed that the obstacle detection algorithm was operating effectively. However, in the current implementation, the obstacle detection algorithm works excessively for vegetation on the outside of the driving area. In future works, the detection algorithm should work within the driving area of the dump truck.

5 Conclusions

In this research, the authors proposed a method of robotization of a conventional six-wheeled dump truck with simple modifications and performed some initial experiments. This research involved the (1) proposal of a retrofitted control module to robotize a dump truck, (2) development of an internal sensor module to enable path tracking motion by the dump truck, (3) derivation of the six-wheeled dump truck kinematics and implementation of path tracking method, and (4) obstacle detection with a 3D LiDAR.

Based on the initial field experiments, the following results were obtained:

1. It was confirmed that the retrofitted robot operated the accelerator, brake, and steering of the dump truck.
2. The integrated system enabled the path tracking motion by vehicle-mounted sensors that detected the vehicle's speed, heading, and articulate angle.
3. Under flat road conditions, the maximum path tracking error was 0.5 m for forward path tracking (6[km/h]) and 1.0 m for backward path tracking (4 km/h).
4. During the autonomous driving (6 km/h) of the dump truck, the obstacle detection method with the 3D LiDAR detected an obstacle (1.5 m high, 0.55 m wide); when the dump truck entered the emergency stop region, the vehicle stopped immediately.

Based on the above, it is possible to perform autonomous driving and enable an emergency stop by applying the proposed method to the conventional six-wheeled dump truck.

In future works, to apply the autonomous dump to real construction sites, the authors plan to implement a combination of functions, such as gear change, forward/backward path tracking, and path planner. Further, the performance evaluation of path tracking and obstacle detection in different vehicle speeds are potential future works.

Acknowledgements This paper is based on the results obtained from a project commissioned by New Energy and Industrial Technology Development Organization (NEDO).

References

1. Hashimoto, Ryou: Promotion consortium (special feature: further promotion of i-construction). Public Works Manag. J. **469**, 8–13 (2017)
2. Andrew, D.: Komatsu, smart construction, creative destruction, and Japan's robot revolution. Asia-Pacific J. **13**(8), 22 (2015)
3. Miura, S., Kuronuma, I., Hamamoto, K.: Next generation construction production system: on automated construction machinery. In: Civil Engineering Conference in the Asian Region, 2016
4. Yin, Y., Rakheja, S., Boileau, P.E: Multi-performance analyses and design optimisation of hydro-pneumatic suspension system for an articulated frame-steered vehicle. Vehicle System Dynamics, pp. 108–133, 2018
5. Meng, Qingyong., Gao, Lulu., Xie, Heping, Dou, Fengqian: Analysis of the dynamic modeling method of articulated vehicles. J. Eng. Sci. Technol. Rev. **10**, 18–27 (2017)
6. Kowatech. Remote steering robot "SAM"
7. Komatsu, T., Okuyama, A.: Integrated design of pressure and position control systems in pneumatic artificial rubber muscle. In: Proceedings of the 58th Japan joint Automatic Control Conference CD-ROM, pp. 113–3, 2015
8. Hemisphere. ssV-102 smart compass. http://www.hemgps.com/ssv102.html
9. Mutoh. Linear encoder. https://www.mutoh.co.jp/products/~deji/linearencoder/index.html
10. Coulter, R.C.: Implementation of the pure pursuit path tracking algorithm. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992
11. Ohta, H., Akai, N., Takeuchi, E., Kato, S., Edahiro, M.: Pure pursuit revisited: field testing of autonomous vehicles in urban areas. In: 2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), pp. 7–12, 2016
12. Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E. and How, J.: Motion planning in complex environments using closed-loop prediction. In: AIAA Guidance,Navigation and Control Conference and Exhibit (2008)
13. Shang, Erke., An, Xiangjing., Tao, Wu., Tingbo, Hu., Yuan, Qiping, He, Hangen: Lidar based negative obstacle detection for field autonomous land vehicles. J. Field Robot. **33**(5), 591–617 (2016)
14. Velodyne LiDAR. Puck, vlp-16. https://velodynelidar.com/vlp-16.html
15. Komatsu. HM300-5. https://www.komatsu.eu/en/dump-trucks/articulated-dump-trucks/hm300-5

Pre-robotic Navigation Identification of Pedestrian Crossings and Their Orientations

Ahmed Farid and Takafumi Matsumaru

1 Introduction

The role of mobile robotics is bound to be more apparent in the future across many fields, such as human assistance, package delivery, and surveillance. One important setting for such robots' operation is pedestrian environments. The annual Tsukuba Challenge held in Japan [1] addresses problems and tasks in such pedestrian settings using mobile robots. In such context, one important task is that of street crossing; robots must be able to recognize street crossings [2, 3] and to determine the safety of crossing [4] during on-line operation.

We argue for the need to identify the locations of street crossings prior to navigation (i.e. prior to on-line operation). Consider the two cases in Fig. 1; assuming a mobile robot with a controller logic for reducing the distance between the robot and a destination, in both cases (a) and (b) the options for motion would increase that distance. Case (a) can be solved if the controller logic was modified, so that the robot can use explicitly marked zebra crossings at the same street intersection to eventually move toward the destination. This would also eventually solve case (b), but no prior knowledge would mean there is a probability that the robot would take a longer route. What we wish to highlight here is that prior knowledge of the environment affects both the controller logic of a robot and time cost to reach a destination. Task efficiency and reducing energy/time costs are especially important for the applications of package delivery and the guidance of visually impaired people.

In this paper, we describe a methodology to identify zebra-pattern street crossings pre-navigation to benefit map construction and path planning purposes. Specifically, we propose a method to identify a zebra crossing's location and orientation in the world frame (i.e., GPS-wise). We are interested in the paradigm of using 2D inter-

---

A. Farid (✉) · T. Matsumaru
Graduate School of Information, Production & Systems, Waseda University, 2-7 Kitakyushu, Fukuoka, Japan
e-mail: ahmed.farid@asagi.waseda.jp

T. Matsumaru
e-mail: matsumaru@waseda.jp

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_6

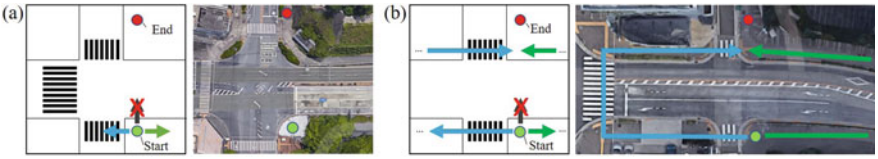


Fig. 1 Two cases with their real-life equivalents in which there was no direct street crossing to the goal. This is to demonstrate the problems of controller logic and time cost to reach a given destination (Satellite images are from Google Maps©)

net maps (e.g., Google Maps, OpenStreet Maps) instead of on-line map recording techniques such as Simultaneous Localization and Mapping (i.e., SLAM) and Structure from Mapping (i.e., SfM) algorithms. In this methodology, we utilize satellite imagery of street intersections because not all 2D digital maps have street crossing information. For the identification process, we trained an object detector neural network on a self-prepared dataset of zebra crossings. We will introduce two methods for verifying the correctness of a zebra crossing detection, followed by an algorithm to estimate the orientation. We estimate orientation as a means of identifying both possible path trajectories and explicit linkages between one city block to another.

2 Related Works

For robots to be able to create path plans and navigate in such environments, a topological understanding, in other words a map, is required. One method suggested by the literature is by using on-line map recording techniques such as SLAM [5, 6] and SfM [7, 8]. Kummerle et al. [5] demonstrated a mobile robot utilizing a SLAM-based 3D map for navigation in a densely populated area. Although some implementations can detect the presence and location of street crossings, time and effort are needed for the on-line recording trips made by robots. Additionally, such robot-recorded maps are not widely available (i.e., no global scale coverage), which means that the preparation of such maps demands time and effort to provide map coverage globally.

To overcome the global map availability problem, research has been made into navigation utilizing 2D internet maps. Hentschel and Wagner [9] utilized OpenStreet Maps to implement path planning and navigation by an autonomous vehicle. Their system, however, depended on the accuracy of map-encoded waypoints. Irie et al. [10] implemented localization by applying edge matching between a 2D map and a real-world RGB camera view. In a similar manner, other research [11, 12] implemented localization but by using 3D point clouds instead. Hosoda et al. [13] devised a technique to generate a simplified map structure by using an input Google Maps image. In the aforementioned works, although a mobile robot was utilized as a testing platform, the locations of street crossings were not explicitly considered; a general

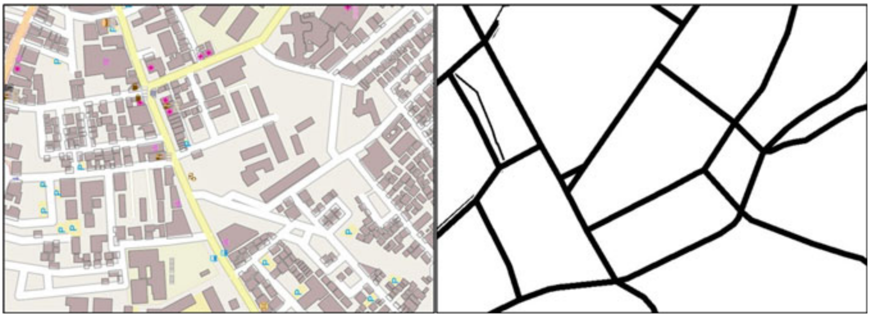


Fig. 2 An example of an urban map converted to a binarized image form. This was done by using the Maperitive software tool [17]

practical problem with 2D internet maps is that explicit data on street crossings is not always available. This problem affected our own previous work [14, 15]; we developed an off-line path planning framework to attempt addressing sidewalk trajectories and street crossing maneuvers, but the accuracy of some test cases was influenced by a missing street crossing information in the utilized maps.

As such, we decided to address the identification of possible locations for street crossings. This is to improve path planning accuracy and overcome the prior knowledge challenges previously mentioned in Sect. 1. The identification process was designed to utilize 2D internet maps for their wide availability and for not requiring on-line map recording by a robot beforehand. Owing to the fact that 2D internet maps do not always have such street crossing information in practice, locations of street intersections were utilized to directly obtain satellite/aerial imagery of such intersections, from which zebra-pattern crossings can be visually detected by a trained neural network (i.e., if not visually obstructed by a bridge or overpass). The output of the proposed work is the GPS-wise locations street crossings, which can be combined with previous works [10, 15, 16] to enhance localization and navigation of mobile robots in pedestrian settings.

3 Methodology Description

3.1 *Sourcing Street Maps and Satellite Imagery*

Map retrieval, in principle, requires the knowledge of its GPS coordinate bounds. In our implementation, we used GPS bounds for desired locations to download OSM map files from OpenStreet Maps. The OSM file[1] contains descriptions of road networks and geometries of buildings. Using such descriptions, we extracted

---

[1] Further description of an OSM file's structure can be found in [9].

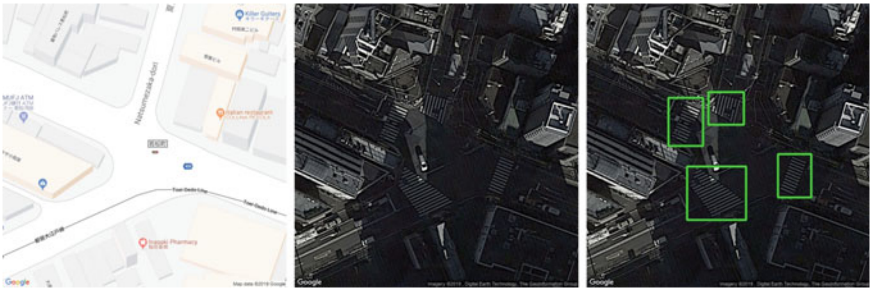


Fig. 3 Zebra crossing detection at a given intersection. Left is a Google Maps© image of the intersection, showing only a traffic signal. Middle is a satellite image of the intersection. Right shows zebra crossings detected by the trained neural network

two important elements: (1) a list of all road intersections occurring in the map, and (2) a graphical representation of the road networks. In our implementation, we utilized the Maperitive software [17] to generate custom-styled maps; in Fig. 2, an example stylized map is shown, which would be later utilized in Sect. 3.2.3. The list of intersections, comprised of GPS coordinates for street intersections, were then used to download satellite imagery directly using Google Maps' API. The images are close-ups of street intersections, which would then act as a source for detecting the location of zebra-pattern crossings.

3.2 Zebra Crossing Detection

3.2.1 Training of an Object Detection Neural Network

As mentioned previously in Sect. 2, information on street crossings is not always present within downloadable maps in practice. Despite maps showing a traffic signal icon as in Fig. 3 (left), it is not indicative of exact locations of street crossings. By utilizing satellite imagery, their existence and locations can be identified. To implement this, we trained an object detection neural network to detect zebra-pattern crossings within a given image. We prepared a dataset of satellite imagery, which includes labels for the locations of zebra crossings. Table 1 provides a summary of the properties of the dataset, as well as the performance of the trained neural network.

Object detectors only provide location information within the pixel space of an input image, but not within the world frame (i.e., GPS). To obtain GPS-wise locations, a mapping from the local pixel space of a given satellite image to the global GPS space is needed. Given the GPS-wise[2] and pixel-wise bounds of an input image,

---

[2] Using the resolution of an image, the aerial zoom level and the GPS center (i.e., the GPS location of a given street intersection), estimation of the GPS bounds is achievable by means of Mercator projection calculations [18].

Table 1 Object detector properties

Property	Value
Training set	115 satellite images (435 crossings)
Testing set	40 satellite images (115 crossings)
Image size	640 × 640
Augmentation	Rotation, flipping, and random darkening
Neural network	Fast Inception V2
Batch size	1
Learning rate	10^{-6}
Training steps	50,000 steps
Graphics hardware	Nvidia GTX1080Ti
Evaluation accuracy (mAP)	89.792%

Table 2 Abbreviations for GPS coordinates-to-pixel-space functions Eqs. 1 and 2

Property	Symbol
Image width	I_w
Image height	I_h
Street crossing longitude	C_{lng}
Street crossing latitude	C_{lat}
X image pixel position of a street crossing	C_x
Y image pixel position of a street crossing	C_y
Left longitude of bounding box	$box_{left,lng}$
Right longitude of bounding box	$box_{right,lng}$
Top latitude of bounding box	$box_{top,lat}$
Bottom latitude of bounding box	$box_{bottom,lat}$

mapping functions to convert pixel positions to GPS latitudes and longitudes were utilized as shown in Eqs. 1 and 2. Equations 3 and 4 are the inverse of 1 and 2 (i.e., conversion from GPS space to pixel space). Table 2 explains the abbreviations of Eqs. 1 and 2.

$$C_{lng} = \frac{C_x * (box_{right,lng} - box_{left,lng})}{I_w} + box_{left,lng} \tag{1}$$

$$C_{lat} = \frac{(I_h - C_y) * (box_{top,lat} - box_{bottom,lat})}{I_h} + box_{bottom,lat} \tag{2}$$

$$C_x = I_w * \frac{C_{lng} - box_{left,lng}}{box_{right,lng} - box_{left,lng}} \tag{3}$$

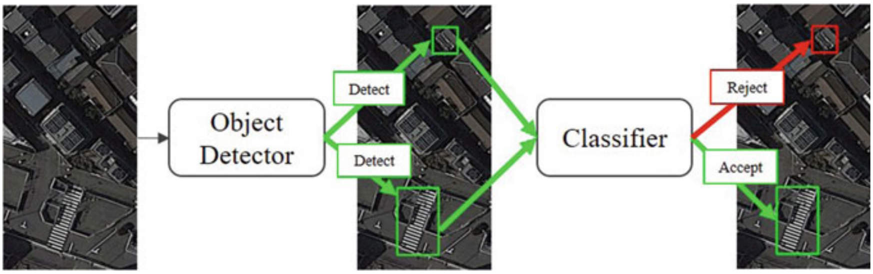


Fig. 4 Illustration of the classifier neural network based verification approach. Satellite imagery was obtained from Google Maps©

Table 3 Classifier properties

Property	Value
Dataset size	474 positives, 65 negatives
Train/Test split	70/30
Augmentation	Rotation, flipping, and random darkening
Neural network	Inception V3 RCNN
Batch size	100
Learning rate	10^{-6}
Training steps	15000 steps
Graphics hardware	Nvidia RTX Titan
Evaluation accuracy	92%

$$C_y = I_h - \frac{I_h * (C_{lat} - box_{bottom,lat})}{box_{top,lat} - box_{bottom,lat}} \tag{4}$$

The trained neural network showed great performance in detecting zebra-pattern crossings. However, there were cases where false positives were detected (Fig. 4). Due to the critical nature of street crossings in a real-world scenario, very high accuracy is desired. In the next two subsections, we will introduce two output verification techniques and will afterward present the evaluated performance.

3.2.2 Classifier Neural Network Based Verification

One idea for verification is to feed the output of the object detector into a classifier neural network; we utilized the fact that the classifier was trained not only on zebra crossings but also on examples of true negatives. In essence, the output of the classifier, based on a confidence threshold, would be binary (e.g., whether a zebra crossing is true or not). An illustration is shown in Fig. 4, training and evaluation properties are shown in Table 3.

3.2.3 Location-Wise Algorithmic Verification

We present an alternative algorithmic method to verify the output of the trained object detector. The reason is that zebra-pattern crossings have two real-life properties that can be utilized: (1) they exist on a street, and (2) they exist between two city blocks (i.e., pedestrians cross from one city block to another). Utilizing these properties, we designed an algorithm that received the positions of the detected crossings, and then calculated the pixel-wise distances between these positions and surrounding city blocks in a map. The city blocks were obtained by applying contour detection on binary map images as similarly retrieved in Sect. 3.1. To achieve this, the GPS positions of detected crossings are mapped into the pixel-space of the binary map image using Eqs. 3 and 4.

Algorithm 1 Location-wise algorithm for zebra crossing detection verification

```
for each zebra crossing: $z$ do
    initialize distances array $d$
    for each city block touching main roads: $b$ do
        $distance$ = get distance between $z$ and $b$
        if $distance$ <road width then
            $d$.append($b$,$distance$)
        end if
    end for
    sort $d$ based on distance index
    if $d$ has more than one element then
        $block1$ = block index of the first element of $d$
        $block2$ = block index of the second element of $d$
        add $z$ to the list of valid zebra crossings
    end if
end for
```

3.2.4 Accuracy of Detection Verification Approaches

In this subsection, accuracy of detection verification approaches is presented. As an experiment, a map for an urban location was used as a test example.[3] By scanning through this map at each street intersection, the trained detector made a total of 82 detections at a set confidence threshold 0.7. Within those 82 detections, there were 76 true positives and 6 true negatives. In Fig. 5, the precision and recall of both the classifier (at thresholds ranging between 0.1 and 0.9) and algorithmic-based approaches are shown.

---

[3] The urban location on which the test was conducted is a random area within the Shinjuku ward in Tokyo, Japan. This area is bound between the GPS coordinates (i.e. latitude and longitude) of [35.6994726204,139.718178435] and [35.6975273796,139.716821565].

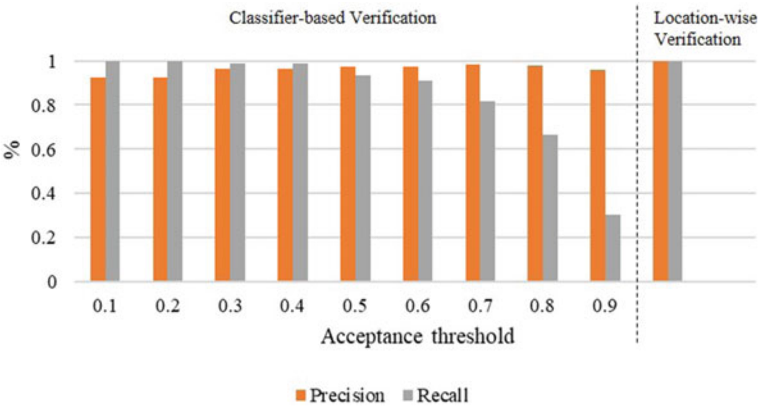


Fig. 5 Precision and recall results for the classifier-based (left) and location-wise algorithmic (right) verification approaches

Although the classifier achieved relatively high accuracy, the algorithmic approach was better. This is owing to the positional properties of zebra-pattern crossings that were utilized by the algorithmic approach. On the other hand, the classifier approach largely depended on its trained dataset for its ability to differentiate between true zebra crossings and false ones. As a side note, the high precision in the graph is indicative of the good ability to find true positives by the trained object detector. Our main motivation from the comparison between classifier-based and algorithmic-based approaches is to demonstrate that, depending on the context and problem setting, not all problems are best solved solely by neural networks.

3.3 Orientation Estimation

3.3.1 Estimation Approach

A technique for estimating orientations was developed to identify path trajectories, which would be useful in linking city blocks to one another in a map construction application. As shown in Fig. 6, orientation estimation is motivated by the presence of street crossings that are diagonal in trajectory (i.e., with respect to the street). Such street crossings, if identified prior to navigation, can be utilized in a path planning process to provide shorter paths.

The main idea of the proposed technique is to attempt to fit a line between zebra stripes, then obtaining the tangent angle of that line to estimate the orientation of a given street crossing. In our proposed technique, the first step was to convert the input zebra crossing image to grayscale, then binarizing it; instead of using a simple threshold function, we utilized K-means clustering ($k = 2$) to binarize. This was

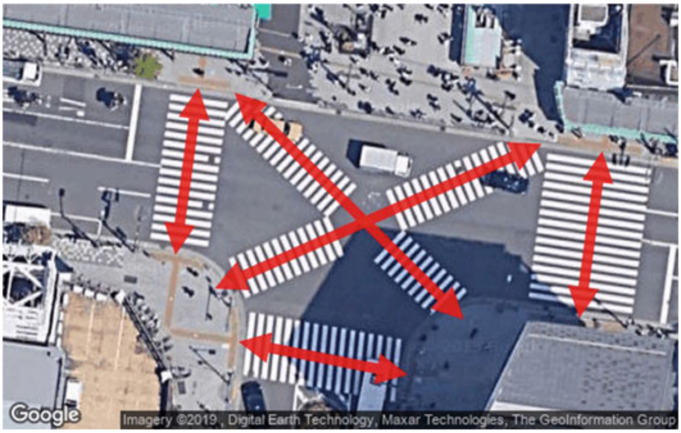

Google
Imagery ©2019, Digital Earth Technology, Maxar Technologies, The GeoInformation Group

Fig. 6 An example satellite image from Google Maps© showing that some streets have diagonal street crossings

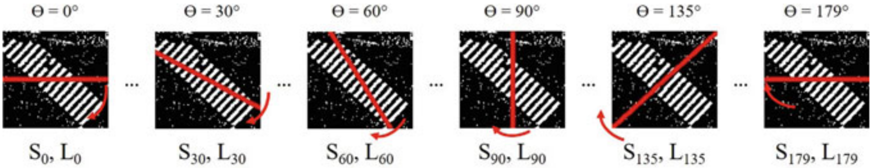

Θ = 0°
Θ = 30°
Θ = 60°
Θ = 90°
Θ = 135°
Θ = 179°
…
S0, L0
S30, L30
S60, L60
S90, L90
S135, L135
S179, L179

Fig. 7 Example of the rotating line scan approach for orientation estimation. Starting from $\theta = 0°$, a line is rotated until it reaches $\theta = 179°$ (assuming 1° resolution for clockwise rotation). Per each angle rotation, the number of zebra stripes intersected (S) and the total intersected length (L) are measured

done because of lighting conditions on a given zebra crossing; owing to the real-world aspect of satellite images captured during the daytime, some zebra crossings can be partially or fully shaded by a nearby building. As a result, we used K-means clustering to implicitly determine the binarization threshold by dividing the bright colors (e.g., the white stripes) from the dark street color. Next, a line was drawn across the binarized image and was rotated in a range from 0° to 179°, assuming 1° rotation resolution. Per each degree rotation, two properties were measured: the number of zebra stripes intersected (S) and the total intersected length (L). As such, we form a 180 × 2 matrix of these measurements as in Eq. 5. An example of the line scan approach is shown in Fig. 7.

$$Score = \begin{bmatrix} S_0, L_0 \\ S_1, L_1 \\ \vdots \\ S_{179}, L_{179} \end{bmatrix} \tag{5}$$

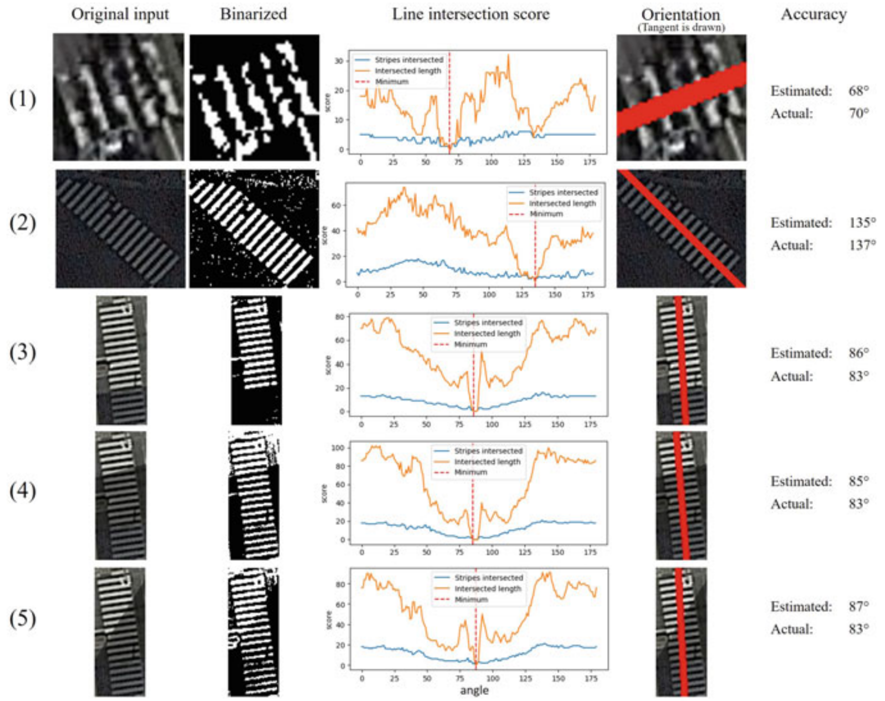


Fig. 8 Exemplary results on orientation estimation for detected zebra crossings. Conditions vary based on real-life fading of the stripes and lighting conditions. Original input zebra crossings were originally from satellite imagery from Google Maps©

Within this matrix, we searched for a minimum combination; we prioritized the least number of stripes intersected, then followed by the intersected length. Upon finding the minimum, the row at which this minimum was found would denote the angle of the fitted line, from which the tangent angle is computed (i.e., fitted line angle plus 90°). This tangent angle was considered the orientation angle of the zebra street crossing.

3.3.2 Exemplary Results for Orientation Estimation

Orientation estimation for five cases of zebra crossings is shown in Fig. 8. The five cases have different properties and thus were used to demonstrate the estimation technique under different circumstances: case (1) is partially faded, case (2) is completely shaded, and case (3) is partially shaded. Cases (4) and (5) were artificially shaded to experiment with different shading ratios and patterns. In all the five cases, the approach managed to estimate the general direction of the input zebra crossing, with the most error difference being 4°. Although such error gap is relatively small, more work is needed to improve upon the accuracy of the algorithm.

4 Conclusion

In this paper, a methodology for identifying zebra-pattern street crossings prior to robotic navigation was demonstrated. Given the relatively high accuracy, we believe that the next step is to integrate the presented methodology into other works (e.g., robot localization/navigation). However, we find room for further improvement of the orientation estimation algorithm to reduce the error margin further. Additionally, we envision an application in which street crossing locations can be identified by an artificial agent, then automatically stored in an internet database (e.g. OpenStreet Maps). This way, not only robots can benefit from understanding street crossing ability prior to navigation, but also humans with visual impairments can be guided by either a smartphone application or even a mobile guidance robot that utilize internet maps.

Acknowledgements This study was supported by Waseda University Grant for Special Research Projects Number 2018A-047. The study was also partially supported by Waseda University-Graduate Program for Embodiment Informatics Research Grant. For both grants, we wish to express our sincere gratitude.

References

1. Tsukuba challenge. Web page http://www.tsukubachallenge.jp/. Accessed March 2019
2. Chand, A., Yuta, S.: Navigation strategy and path planning for autonomous road crossing by outdoor mobile robots. In: 2011 15th International Conference on Advanced Robotics (ICAR) (2011)
3. Chand, A., Yuta, S.: Road-crossing landmarks detection by outdoor mobile robots. J. Robot. Mechatron. **22**(6), 708–717 (2010)
4. Radwan, N., Winterhalter, W., Dornhege, C., Burgard, W.: Why did the robot cross the road? Learning from multi-modal sensor data for autonomous road crossing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017)
5. Kummerle, R., Ruhnke, M., Steder, B., Stachniss, C., Burgard, W.: Autonomous robot navigation in highly populated pedestrian zones. J. Field Robot. **32**(4), 565–589 (2015)
6. Carballo, A., Seiya, S., Lambert, J., Darweesh, H., Narksri, P., Morales, L., Akai, N., Takeuchi, E., Takeda, K.: End-to-end autonomous mobile robot navigation with model-based system support. J. Robot. Mechatron. **30**(4), 563–583 (2018)
7. Song, S., Chandraker, M., Guest, C.: High accuracy monocular SFM and scale correction for autonomous driving. IEEE Trans. Pattern Analy. Machine Intell. **38**(4), 730–743 (2016)
8. Suraj, M., Grimmett, H., Platinsky, L., Ondruska, P.: Predicting trajectories of vehicles using large-scale motion priors. In: 2018 IEEE Intelligent Vehicles Symposium (IV) (2018)
9. Hentschel, M., Wagner, B.: Autonomous robot navigation based on OpenStreetMap geodata. In: 13th International IEEE Conference on Intelligent Transportation Systems (2010)

10. Irie, K., Sugiyama, M., Tomono, M.: A dependence maximization approach towards street map-based localization. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015)
11. Landsiedel, C., Wollherr, D.: Global localization of 3D point clouds in building outline maps of urban outdoor environments. Int. J. Intell. Robot. Appl. **1**(4), 429–441 (2017)
12. Floros, G., van der Zander, B., Leibe, B.: OpenStreetSLAM: global vehicle localization using OpenStreetMaps. In: 2013 IEEE International Conference on Robotics and Automation (2013)
13. Hosoda, Y., Sawahashi, R., Machinaka, N., Yamazaki, R., Sadakuni, Y., Onda, K., Kusakari, R., Kimba, M., Oishi, T., Kuroda, Y.: Robust road-following navigation system with a simple map. J. Robot. Mechatron. **30**(4), 552–562 (2018)
14. Farid, A., Matsumaru, T.: Path planning of sidewalks & street crossings in pedestrian environments using 2D map visual inference. In: ROMANSY 22 Robot Design, Dynamics and Control, pp. 247–255 (2018)
15. Farid, A., Matsumaru, T.: Path planning in outdoor pedestrian settings using 2D digital maps. J. Robot. Mechatron. **31**(3), 464–473 (2019)
16. Suzuki, T., Kitamura, M., Amano, Y., Kubo, N.: Autonomous navigation of a mobile robot based on GNSS/DR integration in outdoor environments. J. Robot. Mechatron. **26**(2), 214–224 (2014)
17. Maperitive. Web page. http://maperitive.net/
18. Olson, E.: Autonomous Robotics Laboratory Lecture Notes, "Map Projections and GPS", EECS467. University of Michigan, Winter. https://april.eecs.umich.edu/courses/eecs467_w14/wiki/images/5/5e/Maps_gps.pdf (2014)

Improvement in Measurement Area of Three-Dimensional LiDAR Using Mirrors Mounted on Mobile Robots

Kazuki Matsubara and Keiji Nagatani

1 Introduction

In recent years, the demand for automation of plant inspection using robot technology has increased. Reducing labor and increasing safety are important reasons behind this demand. Particularly, companies having offshore platforms expect to realize mobile robots for an autonomous inspection, because the burden on workers and the cost of labor are high [9]. Tracked vehicles are attracting attention as an option to achieve such tasks. They exhibit good traversal ability on steps/stairs and high loading capacity in installing various sensors. When traveling autonomously, this type of mobile robot is required to generate an environmental map and/or to localize itself. Light detection and ranging (LiDAR) is often used to acquire information regarding the surrounding environment. Notably, an omnidirectional three-dimensional LiDAR (3D-LiDAR) that can acquire the 3D point cloud of the surrounding environment has become popular recently.

In general, omnidirectional 3D-LiDAR should be installed on the top of a mobile robot to measure omnidirectionally. However, when LiDAR is installed at the top of mobile robots, as shown in Fig. 1a, several problems arise. For example, the overall undetectable areas close to the robot increases, and the LiDAR may block other sensors' sensing area. The former issue is serious because of the risk of collision with obstacles or the increased possibility of the robot falling down from steps/stairs.

Hence, for expanding the measurable area, swinging the LiDAR itself [6] or installing multiple LiDARs are considered. However, mounting additional swinging functions or increasing the number of sensors is undesirable in terms of the weight and cost of the robot.

In-plant inspection robots, focused by the authors, rarely move backwards. Therefore, an omnidirectional 3D-LiDAR is typically installed at the front of the mobile

---

K. Matsubara (✉)
Tohoku University, 468-1, Aza-Aoba Aramaki, Aoba-ku, Sendai, Japan
e-mail: matsubara@frl.mech.tohoku.ac.jp

K. Nagatani
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: keiji@ieee.org

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_7

robot instead, as shown in Fig. 1b, to avoid the aforementioned problems. However, in this case, the robot body blocks the laser rays behind the sensor, and half of the sensor information that can be acquired originally is invalidated. Additionally, even in such a remaining available area, LiDAR still has a blind spot near the robot as its measurement range in the vertical direction is limited.

Therefore, in this research, to improve the environmental measurement area of LiDAR, the authors focus on the combination of LiDAR and mirrors. Installing mirrors behind LiDAR, as shown in Fig. 1c, enables the invalidated area behind the LiDAR to be utilized. Hence, it is possible to acquire environmental information within a broader range in front of the robot. Particularly, by expanding the measurement area to the lower front of the robot, it is possible to obtain terrain information such as a shape of the stairway or large obstacles in the short distance of the robot [4, 8, 10].

In the previous studies on the combination of LiDAR and mirrors, S. Abiko et al. proposed a method to expand the measurement area of LiDAR by combining a two-dimensional LiDAR and mirrors to enhance the environment-recognition ability of flying robots [1]. Additionally, H. Noguchi et al. proposed a method to obtain a 3D point cloud with high density by combining a two-dimensional LiDAR with mirrors and a rotation mechanism [5]. However, studies that combine 3D-LiDAR and mirrors for inspection robots are non-existent, to the author's best knowledge.

Herein, the installation condition of the mirror is proposed to obtain information regarding the desired measurement area with invalidated laser rays. Subsequently, an experimental report is described to evaluate the validity and accuracy of the proposed method. Furthermore, short experiments with a tracked vehicle are reported to acquire the point cloud in a real environment.

2 Installation Condition of Mirrors to Acquire Desired Detection Areas

2.1 Mirror Installation Requirements

To expand the measurable area of LiDAR by combining mirrors and omnidirectional 3D-LiDAR installed in front of the mobile robot, the following requirements should be satisfied:

- The overlap between the desired area to be observed (hereinafter called "target area") and the detection area by the reflected laser ray (hereinafter called "reflective area") is maximized.
- The overlap between the detection area of standard laser ray (hereinafter called "normal area") and the reflective area is minimized.

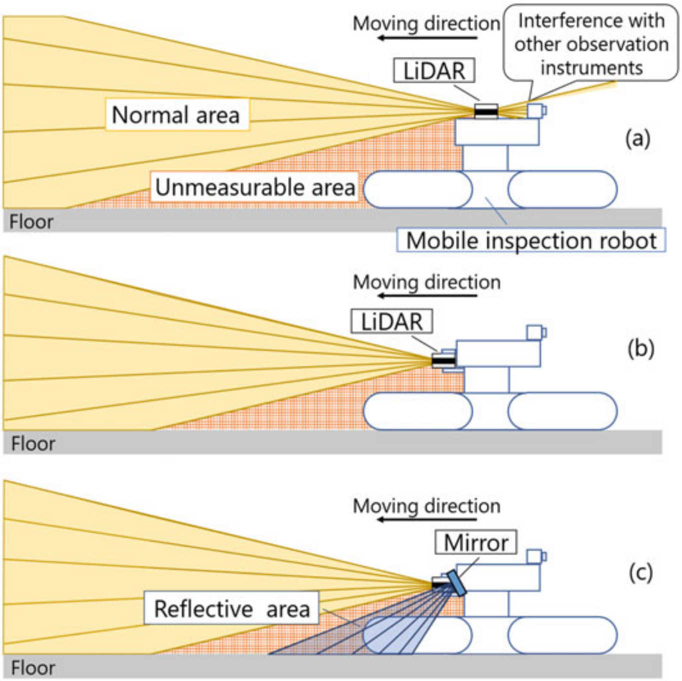
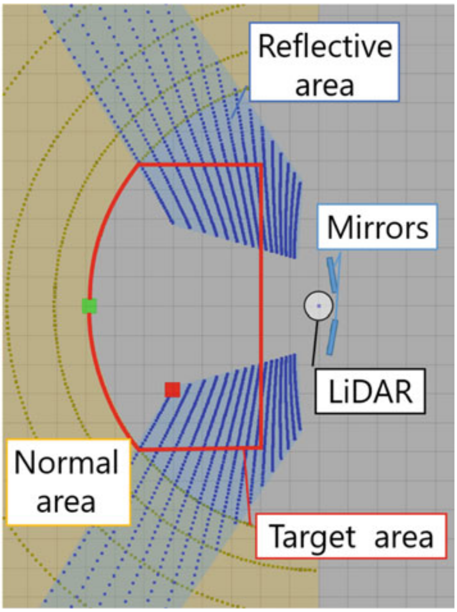

Moving direction
Interference with other observation instruments
LiDAR
Normal area
(a)
Unmeasurable area
Floor
Mobile inspection robot
Moving direction
LiDAR
(b)
Floor
Moving direction
Mirror
(c)
Reflective area
Floor

Fig. 1 Outline drawing of mobile inspection robot and explanation of measurable area of LiDAR: **a** LiDAR installed at the top of the robot, **b** LiDAR installed at the front of the robot, **c** LiDAR installed at the front of the robot with mirrors

Fig. 2 How the laser spread after reflection


Reflective area
Mirrors
LiDAR
Normal area
Target area

Moreover, in this research, the objective is to recognize obstacles at a short distance from the robot or the floor surface condition by expanding the measurable area. Therefore, the gap between the normal area and reflective area should be minimized.

Furthermore, two additional points in determining the installation angle of the mirrors are noteworthy.

- To use flat mirrors to avoid unintended laser ray behaviors due to the distortion of the mirrors' surface or reflection of the laser light.
- To not locate the mirror at a position where the laser reflected by the mirror directly enters LiDAR to avoid LiDAR damage.

Based on the requirements above, the mirror installation angle is determined by solving the geometrical conditions shown in the next section. Note that, mirrors should be installed on the same base with LiDAR to avoid displacement of their surface caused by vehicle vibration during driving.

2.2 Geometrical Conditions to Satisfy Requirements

Figure 2 shows an example of a geometric calculation result of the point cloud acquired using the normal laser ray and the laser ray reflected from mirrors, as shown in Fig. 1c. The yellow area corresponds to the normal area, the blue area corresponds to the reflective area, and the area surrounded by the red frame is the target area. The normal area extends concentrically around the LiDAR. Meanwhile, the reflective area is a quadrilateral where the point density decreases with distance from the center.

As shown in Fig. 3, let the pitch angle α be the irradiation angle in the pitch direction of the laser ray, and let the yaw angle β be the irradiation angle in the yaw direction. The green point in Fig. 2 shows the contact point between the laser ray and the floor, where α is the smallest and β is $0°$ among the normal laser rays. Further, the red point (hereinafter, a laser beam capable of acquiring this point called "target laser") represents a contact point between the laser ray and the floor, where α and β are the two largest among the reflected laser rays. Therefore, when the green point position corresponds with the red point position in Fig. 2, the overlap area between the reflective area and target area is maximized, and the overlap area between the normal area and reflective area is minimized. However, the target laser has to be parallel to the xz plane to satisfy the prerequisite that the reflected laser should not enter the LiDAR directly. Based on the above, to obtain the installation angle of a mirror that satisfies the requirements and the emphasize points in Sect. 2.1, the target laser should fulfill the following geometrical conditions:

1. The target laser satisfies the following conditions.
    - The irradiation angle in the pitch direction is the largest.
    - Laser ray reflected becomes parallel to the xz plane.

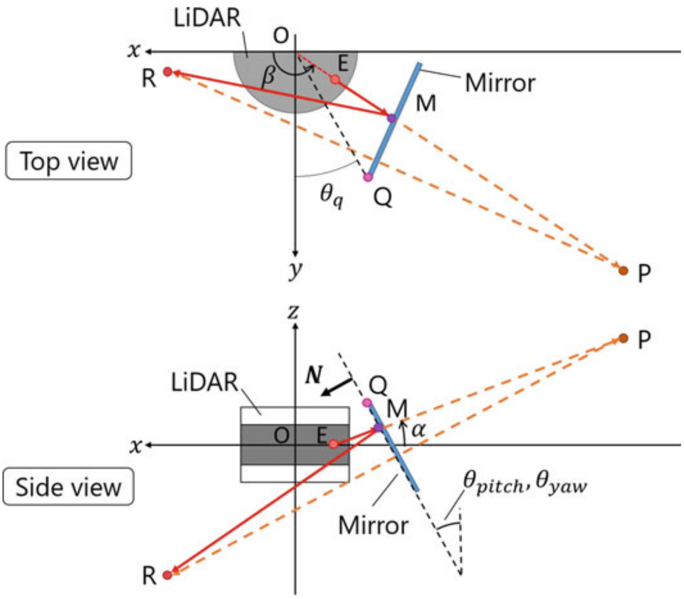


Fig. 3 Geometric conditions for determining mirror installation angle

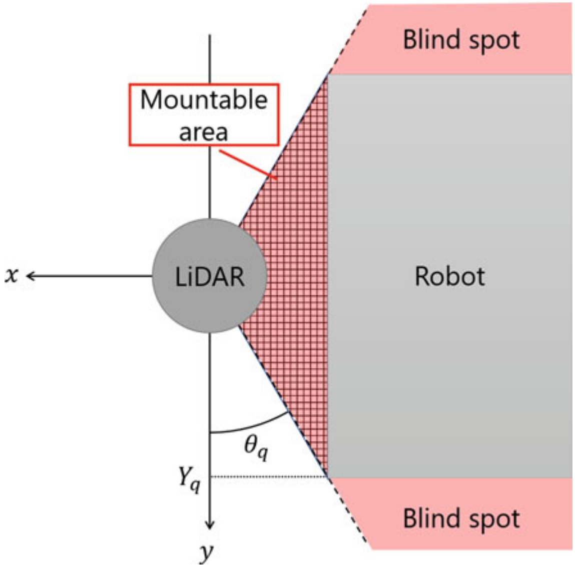
2. Let RV be the distance between the target laser and x-axis. This implies $y_r = RV$.
3. A plane representing a mirror passes through a point Q$= (x_q, y_q, z_q)$ with the following x, y, z coordinates:
   - x_q x coordinate satisfying Y_q, θ_q shown in Fig. 4 in the xy plane,
   - y_q y coordinate with the value of Y_q shown in Fig. 4
   - z_q z coordinate satisfying $x = x_q, y = y_q$ on the laser ray with the largest irradiation angle in the pitch direction.

Figure 3 shows the characters used under the geometric conditions above. Figure 4 shows the area where the mirror can be installed (hereinafter called "mountable area").

2.3 Method of Deriving Mirror Angle that Satisfies Design Requirements

In this section, the installation angle of the mirror is derived from the geometrical condition shown in Sect. 2.2. First, the irradiation angle β in the yaw direction from LiDAR is obtained such that the target laser is parallel to the xz plane after specular reflection. Let M be the point of intersection of the laser ray and the mirror surface,

Fig. 4 Mountable area where mirrors can be installed



$\boldsymbol{N}$ be the normal vector of the mirror surface, and θ_{pitch} and θ_{yaw} be the rotation angles of the mirror's pitch direction and yaw direction, respectively. At this time, the normal vector $\boldsymbol{N}$ of the mirror surface can be described as Equation (1).

$$\boldsymbol{N} = \begin{pmatrix} \cos(\theta_{yaw})\cos(\theta_{pitch}) \\ \sin(\theta_{yaw})\cos(\theta_{pitch}) \\ -\sin(\theta_{pitch}) \end{pmatrix} = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} \tag{1}$$

To create a target laser ray that is parallel to the xz plane after specular reflection, the y coordinates of point M and point R should be RV. The irradiation angle β of the target laser beam in the yaw direction that satisfies this condition can be written as in Eq. (2):

$$\beta = \cos^{-1}\Big(\frac{-(R_v N_x S)(R_v N_z C)}{(R_v N_x S)^2 + \xi^2} - \frac{\sqrt{((R_v N_x S)(R_v N_z C))^2 - ((R_v N_x S)^2 + \xi^2)((R_v N_z C)^2 - \xi^2)}}{(R_v N_x S)^2 + \xi^2}\Big) \tag{2}$$

It is noteworthy that ξ represented by Eq. (3).

$$\xi = N_z EC + (\boldsymbol{N} \cdot \overrightarrow{OQ})S - R_v N_y S \tag{3}$$

Next, obtain the mirror installation angles θ_{pitch} and θ_{yaw}. Let P be the coordinates obtained by the laser beam reflected by LiDAR, and E be the irradiation point of the laser beam. $\overrightarrow{EM}$ can be represented by the two Eqs. (4) and (5) from Fig. 3:

$$\begin{aligned}
&\overrightarrow{EM} \\
&= \overrightarrow{EO} + \overrightarrow{OR} + \overrightarrow{RP} + \overrightarrow{PM} \\
&= (1 + \frac{L}{\boldsymbol{N} \cdot \overrightarrow{OR} - 2L - \boldsymbol{N} \cdot \overrightarrow{OE}})(\overrightarrow{OR} - 2L\boldsymbol{N} - \overrightarrow{OE}) \\
&= \boldsymbol{f}(\theta_{pitch}, \theta_{yaw}),
\end{aligned} \tag{4}$$

$$\overrightarrow{EM} = \frac{\boldsymbol{N} \cdot \overrightarrow{EQ}}{\boldsymbol{N} \cdot \boldsymbol{n}} \boldsymbol{n} = \boldsymbol{g}(\theta_{pitch}, \theta_{yaw}). \tag{5}$$

It is noteworthy that L and $\boldsymbol{n}$ are the value represented by Eq. (6) and the vector represented by Equation (7), respectively.

$$L = |(\overrightarrow{OR} - \overrightarrow{OQ}) \cdot \boldsymbol{N}| \tag{6}$$

$$\boldsymbol{n} = \begin{pmatrix} \sin(\alpha)\cos(\beta) \\ \sin(\alpha)\sin(\beta) \\ \cos(\alpha) \end{pmatrix} \tag{7}$$

Here, both $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw})$ and $\boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$ represent the same $\overrightarrow{EM}$. However, depending on the values of θ_{pitch} and θ_{yaw}, $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw})$ does not correspond with $\boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$. $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw})$ and $\boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$ match only when θ_{pitch} and θ_{yaw} satisfy the geometrical conditions given in Sect. 2.2. Hence, the installation angle θ_{pitch} and θ_{yaw} of the mirror is a value that satisfies Eq. (8).

$$\boldsymbol{f}(\theta_{pitch}, \theta_{yaw}) - \boldsymbol{g}(\theta_{pitch}, \theta_{yaw}) = \boldsymbol{0} \tag{8}$$

3 Verification Test

According to the mirror installation requirements described in Sect. 2.2, a verification test was conducted to confirm the validity of the mirror installation requirements and the accuracy of the point cloud in the reflective area.

3.1 Test Method

Figure 5 shows the test environment. First, the consistency between the reflective area and target area was verified by visualizing the point cloud that can be acquired from the laser ray reflected by the mirrors. Next, the target sphere with a height of 15 mm

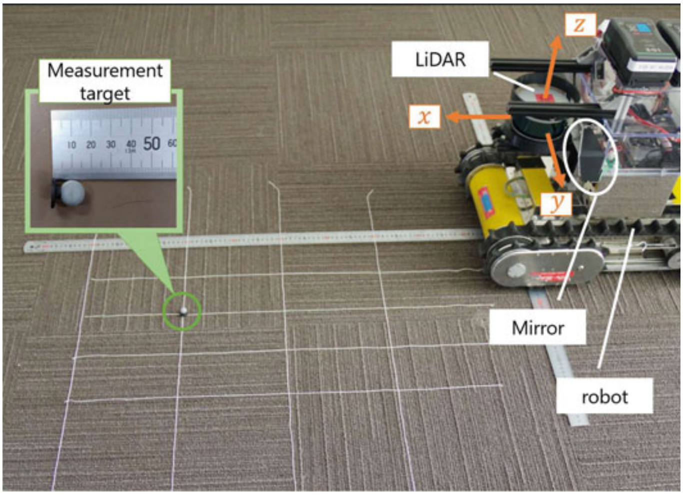


Fig. 5 Test environment

Table 1 Specification of Velodyne VLP-16

Specification	Value
Number of lasers	16
Detection area	(Horizontal) 360°, (Vertical) ± 15°
Measurement error	± 3 cm (1 standard deviation at 25 m)
Installation height	310 mm

and a radius of 5 mm, covered by a highly reflective material, was placed in the target area, as shown in Fig. 5. Subsequently, the measurement accuracy in the reflective area was verified by comparing the target sphere position obtained from the LiDAR and that from a manual. The target sphere was placed in 32 locations, where the coordinates of the measurement target were $x = (300\text{mm}, 500\text{mm}, 700\text{mm}, 900\text{mm})$ $y = (|100\text{mm}|, |200\text{mm}|, |300\text{mm}|, |400\text{mm}|)$. In this test, VLP-16, manufactured by Velodyne, was equipped for LiDAR application. Its primary specification is shown in Table 1. From the LiDAR installation position and the mirror installation angle requirements described in Sect. 2.2, the size of $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw}) - \boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$ becomes that shown in Fig. 7. As shown in Fig. 7, the installation angle of the mirror was determined to be inward for 21.0° around the z-axis and 15.5° around the x-axis in Fig. 3 because the size of $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw}) - \boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$ is the smallest. Figure 8 shows the reflective area calculated from the determined mirror installation requirements (Fig. 6).

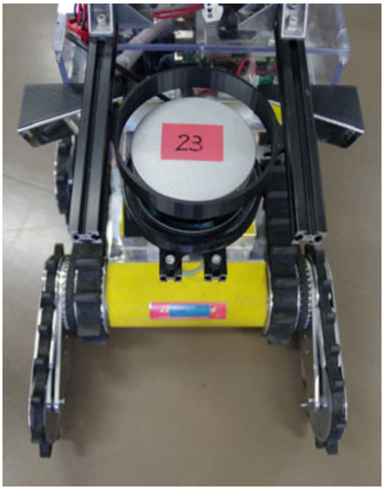


Fig. 6 Enlarged view of the mirror installation part

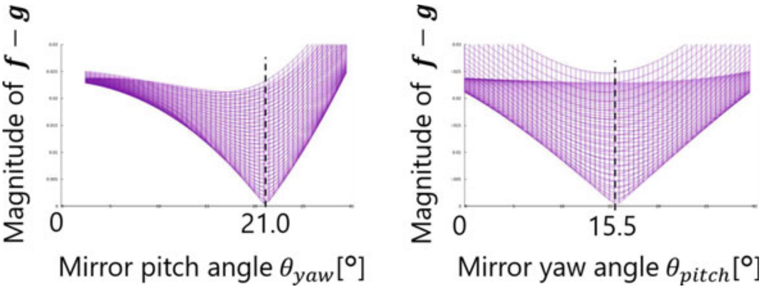


Fig. 7 Magnitude of $\boldsymbol{f}(\theta_{pitch}, \theta_{yaw}) - \boldsymbol{g}(\theta_{pitch}, \theta_{yaw})$ satisfying installation requirements

3.2 Calibration of Reflected Points

To reflect the point cloud obtained from LiDAR through the mirror surface, the normal vector information of the mirror surface and the position vector of the mirror is required. However, because of mirror installation errors, the coordinate transformation may be incorrect and the acquired point cloud in the reflective area will deviate from the actual values. Therefore, in this research, Rviz, a visualization tool for the robot operating system (ROS) [7] was used to correct the deviations of the normal vector of the mirror surface and the position vector. Figure 9 shows the appearance of Rviz while correcting the mounting angle of the mirror. The position and angle of the mirrors, shown as two plates in Rviz, can be changed manually by translating them along the x-axis and rotating them around the y- and z-axes. The calculation of coordinate transformation of the acquisition point cloud in the reflective area can be displayed in real time.

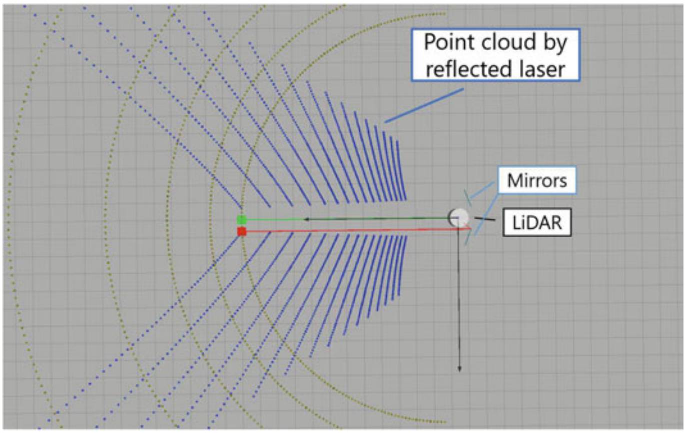
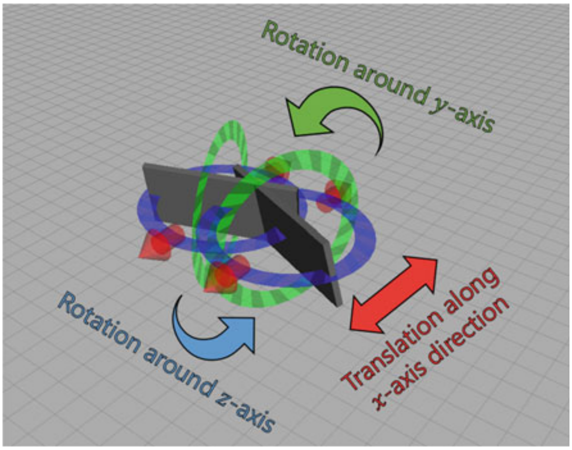

Point cloud by reflected laser
Mirrors
LiDAR

Fig. 8 Expected reflective area from calculation

Fig. 9 Rviz appearance for mounting angle and position adjustment of the mirrors

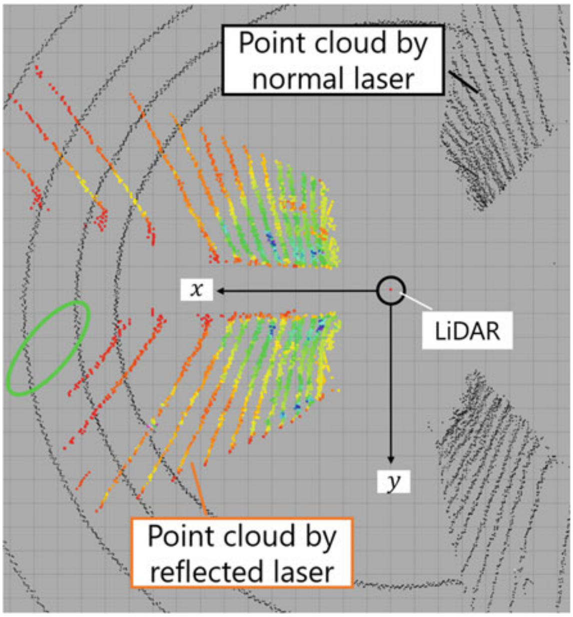

Rotation around y-axis
Rotation around z-axis
Translation along x-axis direction

In this calibration step, the measurement target was placed at a position where both the normal laser and reflective laser can be measured. Subsequently, the correct position and angle of the mirrors were acquired by manually adjusting the mirrors, such that the measurement target obtained by the reflected laser ray corresponds with that obtained by the normal laser ray.

3.3 Results

Figure 10 shows the visualization of the resulting point cloud by Rviz. According to the comparison between Figs. 8 and 10, it was confirmed that the measurement area had expanded as designed. However, for the part surrounded by a green ellipse, the expected point cloud information could not be obtained.

Fig. 10 Rviz visualization of resulting point cloud



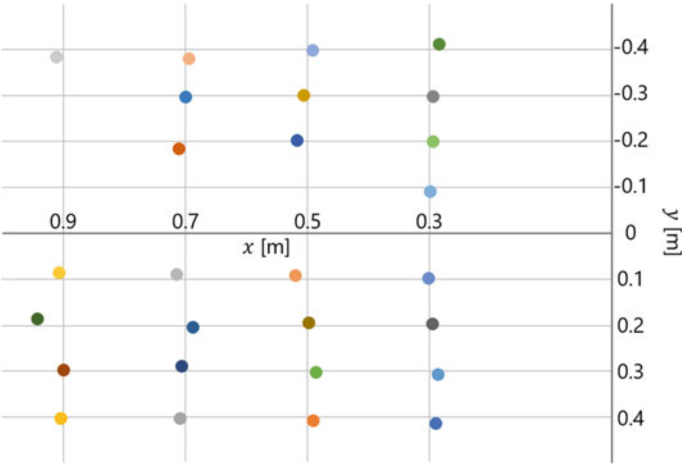


Fig. 11 Position information of detected measurement target projected onto xy plane

Next, Fig. 11 shows the position information of the detected target sphere projected onto the xy plane. Measurement and accuracy evaluation were performed at each intersection of the grid lines of the x and y axes, and the points indicate the average position of the acquired measurement target. The standard deviation of the error between the position information obtained by LiDAR and the measured value was 15 mm in the xy plane and 9 mm in the z direction.

3.4 Discussion

The point cloud that could not be acquired were all due to the laser ray with a large irradiation pitch angle α. Therefore, it is thought that the calculation results and the test results did not match owing to the installation error of the mirror.

Meanwhile, regarding the accuracy of the acquired point cloud, the errors in both the xy plane and z-axis direction are within the measurement error of LiDAR; therefore it can be concluded that the coordinate transformation is performed with sufficient accuracy.

In conclusion, the results indicate that the installation requirements of the mirror and the calculation results of the coordinate transformation are appropriate. Thus, environmental information can be acquired with sufficient accuracy from the point cloud in the reflective area by combining omnidirectional 3D-LiDAR and the mirror installed at the proposed installation angle.

4 Utility Verification Test in the Proposed Method

When a tracked vehicle inspects a plant, stairs are one of the places where movements become particularly hard. It is difficult to climb the stairs because, in the case of a standard LiDAR placement, environmental information cannot be obtained at short-range distances. In this section, we will report the effect of the tracked vehicle on stair climbing when using the proposed LiDAR measurement range expansion method.

4.1 Required Information for a Tracked Vehicle to Climb Stairs

The environmental information related to the stairs, which is necessary for control when a tracked vehicle climbs stairs, includes the following for each scene [2, 3]:

- Start ascending: Slope of the starting step
- During ascend: Stairs' slope
- Finish ascending: Positional relationship between the landing step and the robot
- Start descending: Slope of the landing step
- During descend: Stairs' slope
- Finish descending: Height of the starting step, Stairs' slope.

Among the scenes, the robot with the standard LiDAR installation cannot detect the stairs information of the following three scenes: during ascend, finish ascending, and start descending. Table 2 shows the information required for climbing stairs and current solutions for these scenes.

Table 2 Necessary stairs information and current solutions when climbing stairs

Scene	Necessary stairs information	Current solutions
During ascend	Stairs slope	Measure the stairs' size in advance
Finish ascending	Positional relationship between the landing step and the robot	Place landmarks such as reflectors in the environment Measure the number of stairs in advance and measure the distance by odometry
Start descending	Slope of the landing step Positional relationship between the landing and the robot	Place landmarks such as reflectors in the environment Measure change in IMU information using sub crawlers

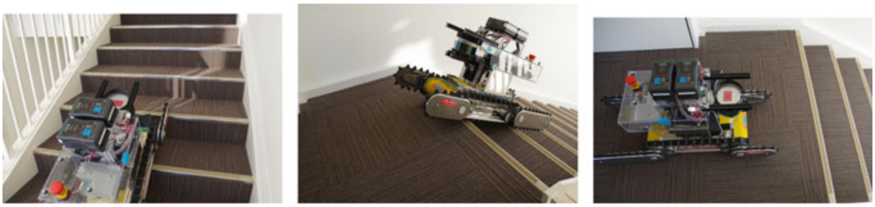
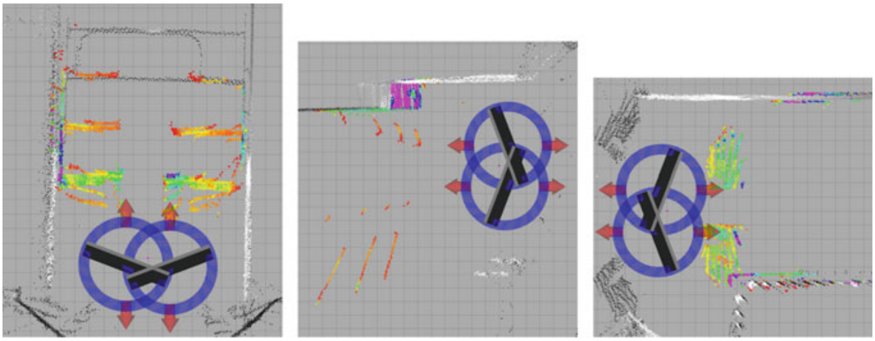

During ascend (b) Finish ascending (c) Start descending

Fig. 12 Stairs environment and robot postures

It is difficult to estimate a robot's status using odometry and inertial measurement unit (IMU) information owing to slips, disturbances, and gyroscope drifts [2]. In addition, although the installation of landmarks is considered as a practical solution, it is not always possible to install them and to not be acquired stably on any stairs. When it is not possible to acquire such stairs information, landing on the landing step may fail, and that may produce a strong impact to the robot, or in the worst case, the robot may fall downstairs. Therefore, by applying the proposed method combining LiDAR and mirrors to a robot, the robot can acquire the stairs information for climbing the stairs in all scenes.

4.2 Test Method

To confirm whether the proposed method is effective for stair climbing by a tracked vehicle, an environment-recognition test was conducted for three real scenes: during ascend, finish ascending, and start descending, using real robots while traversing the stairs. Figure 12 shows the stairs environment in each scene. LiDAR and the position and angle of the mirrors are the same as in Sect. 3.



(a) During ascend (b) Finish ascending (c) Start descending

Fig. 13 Test results near stairs

4.3 Test Results

Figure 13 shows the recognition test results for the three scenes shown in Sect. 4.2.

As shown in Fig. 13a for the "during ascend" scene, it was possible to detect the cross section and edge of the stairs while ascending. According to the result, the robot can calculate the stairs' slope while climbing up the stairs. In addition, because the skirting board can be detected, the robot can estimate its position on the stairs.

As shown in Fig. 13b for the "finish ascending" scene at the end of the stairs, the surface of the landing step was acquired, although the sub-tracks blocked the reflective area detection. According to the result, it is possible to obtain the positional relationship between the landing step and the robot, which is crucial for a robot to land safely on the landing step of the stairs.

As shown in Fig. 13c for the "start descending" scene, the edge of the stairs near the downstairs was detected. In addition, because the skirting boards were detected, the robot can be positioned for "start descending" without setting the landmarks in the environment. Based on the results above, it was confirmed that the robot acquired the necessary environment information for ascending/descending stairs by the additional two mirrors, which was difficult for the typical installation position of LiDAR.

5 Conclusion

In this research, the authors aim to expand the measurement area of LiDAR by additional mirrors behind omnidirectional 3D-LiDAR to improve an ability of acquiring information regarding the front environment. Hence, the design requirements of the mirror were proposed, the accuracy evaluation test of the detected point cloud was conducted, and the usefulness test in the real environment was performed. Although the reflective area was slightly different from the calculation results, it was confirmed that the position information of the obtained point cloud was acquired with practical accuracy.

In future works, based on the proposed method, the autonomy of tracked vehicles will be realized, such as obstacle detection and avoidance while traveling, and ascending/descending stairs with no environment information without any landmarks in advance.

Acknowledgements Mitsubishi Heavy Industries supported this work.

References

1. Abiko, S., Sakamoto, Y., Hasegawa, T., Yuta, S., Shimaji, N.: Development of constant altitude flight system using two dimensional laser range finder with mirrors (2017). https://doi.org/10.1109/AIM.2017.8014121
2. Endo, D., Watanabe, A., Nagatani, K.: Stair climbing control for 4-DOF tracked vehicle based on internal sensors. J. Robot. (2017). https://doi.org/10.1109/SSRR.2016.7784286
3. Guo, J., Shi, J., Zhu, W., Wang, J.: Approach to autonomous stair climbing for tracked robot. In: 2017 IEEE International Conference on Unmanned Systems (ICUS), pp. 182–186 (2017). https://doi.org/10.1109/ICUS.2017.8278337
4. Lee, H., Chung, W.: Terrain classification for mobile robots on the basis of support vector data description. Int. J. Precision Eng. Manufact. **19**(9), 1305–1315 (2018). https://doi.org/10.1007/s12541-018-0154-4
5. Noguchi, H., Handa, M., Fukui, R., Mori, T., Sato, T., Sanada, H.: Capturing device for dense point cloud of indoor people using horizontal LIDAR and pan rotation of vertical LIDAR with mirrors. In: IEEE/SICE International Symposium on System Integration (SII), pp. 428–433 (2012)
6. Ohno, K., Tadokoro, S., Nagatani, K., Koyanagi, E., Yoshida, T.: Trials of 3-D map construction using the tele-operated tracked vehicle kenaf at disaster city. In: 2010 IEEE International Conference on Robotics and Automation, pp. 2864–2870 (2010). https://doi.org/10.1109/ROBOT.2010.5509722
7. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5. Kobe, Japan (2009)
8. Sinha, A., Papadakis, P.: Mind the gap: detection and traversability analysis of terrain gaps using LIDAR for safe robot navigation. Robotica **31**, 1085–1101 (2013). https://doi.org/10.1017/S0263574713000349
9. Transport Ministry of Land I, (MLIT) T (2014) Efforts to take a far-sighted vision of social infrastructures. http://www.mlit.go.jp/common/001063080.pdf
10. Westfechtel, T., Ohno, K., Mertsching, B., Hamada, R., Nickchen, D., Kojima, S., Tadokoro, S.: Robust stairway-detection and localization method for mobile robots using a graph-based model and competing initializations. Int. J. Robot. Res. **37**(12), 1463–1483 (2018). https://doi.org/10.1177/0278364918798039

The Hulk: Design and Development of a Weather-Proof Vehicle for Long-Term Autonomy in Outdoor Environments

Stephen Kyberd, Jonathan Attias, Peter Get, Paul Murcutt, Chris Prahacs, Matthew Towlson, Simon Venn, Andreia Vasconcelos, Matthew Gadd, Daniele De Martini, and Paul Newman

1 Introduction

There is a large and growing need for autonomous robots to perform a wide range of practical tasks in the near future. These span a wide range of industries, from autonomous cars and transportation to robots for hazardous environments such as

---

S. Kyberd · J. Attias · P. Get · P. Murcutt · C. Prahacs · M. Towlson · S. Venn · A. Vasconcelos ·
M. Gadd (✉) · D. D. Martini · P. Newman
Oxford Robotics Institute, 23 Banbury Road, Oxford, UK
e-mail: mattgadd@robots.ox.ac.uk

S. Kyberd
e-mail: skyberd@robots.ox.ac.uk

J. Attias
e-mail: jattias@robots.ox.ac.uk

P. Get
e-mail: peterget@robots.ox.ac.uk

P. Murcutt
e-mail: pmurcutt@robots.ox.ac.uk

C. Prahacs
e-mail: cprahacs@robots.ox.ac.uk

M. Towlson
e-mail: matthew@robots.ox.ac.uk

S. Venn
e-mail: svenn@robots.ox.ac.uk

A. Vasconcelos
e-mail: andreia@robots.ox.ac.uk

D. D. Martini
e-mail: daniele@robots.ox.ac.uk

P. Newman
e-mail: pnewman@robots.ox.ac.uk

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_8

oil fields [8]. A plethora of platforms are available for research and development in many of these areas and some are available for commercial deployment. These include mobile indoor robots [18], off-road robots such as those from Clearpath Robotics[1] and even explosion-proof robots [10].

Robots designed to be deployed into an outdoor environment span a wide range of sizes and capabilities. For example from Clearpath they range from the Jackal[2] to the Moose.[3] Due to the simplicity of control almost all outdoor robots currently available use skid steering, but this means that on damp ground, even moderately tight turns are liable to damage the surface. While acceptable for industrial applications or rough terrain use, this is entirely unacceptable for many other applications, such as agriculture or operations on manicured grassed areas. This limits the ability to deploy robots into a wide range of potential applications; including security patrols, transport, and lawn or pitch maintenance.

This paper describes the engineering design principles used to realise an Unmanned Ground Vehicle (UGV) suitable for operation over extended time-scales without expert supervision. We build on our experience in past projects—including the very early Segway—Lisa [17], the offroading Bowler Wildcat [16], the urban Nissan Leaf RobotCar [9], and light transport LUTZ Pathfinder pod [12]—and detail the holistic approach taken in order to meet a set of requirements dictated by a third-party group of clients which require the robot to operate in a challenging, unstructured, grassy environment.

This work is structured as follows. Section 2 delineates the system requirements for the platform, while Sect. 3 describes the process that led to the choice of the base vehicle for the robot. Section 4 and Sect. 5 describe the two main groups of components installed on the platform. The mechanical modifications needed are covered in Sect. 6. Finally, the HRI system installed on the platform is described in Sect. 7.

2 Design Guidelines

When setting out to create the robotic vehicle, a set of extensive design requirements were created that helped to guide the process. We refer to the retrofitted platform that we designed to fulfill these requirements as the Hulk. At a high level, the following requirements are included:

Long-term autonomous operation: The key requirement is that the vehicle should be able to operate for long periods—weeks or months—reliably and unattended, except for being put on charge. It must perform tasks like inspection and transport in diverse and unstructured environments and weather conditions. This influenced the choice of the base vehicle and waterproofing requirements (Sect. 6); as well as the components installed, like the MVC and sensor payload (Sect. 6).

---

[1] Clearpath Robotics Inc.: https://www.clearpathrobotics.com.

[2] Jackal platform: https://bit.ly/2I3JzRP.

[3] Moose platform: https://bit.ly/2uIrgc7.

Terrain capabilities: To enable the widest possible scope for deployment, the vehicle should have little or no impact on the surface it is driving on. This heavily influenced the choice of base vehicle (Sect. 3).

Human safety: The resulting platform is a large and relatively quick vehicle: the safety of humans and objects around it is of primary concern. A hardware and software safety stack is designed (Sect. 5) and bystander interaction considered (Sect. 7).

Simplicity of use: Nontechnical users should be able to operate and work alongside the Hulk with minimal prior knowledge; an HRI system has been added to facilitate this (Sect. 7).

3 Base Vehicle

A key requirement is that the robot is able to operate for weeks or months reliably. A COTS vehicle, with custom modifications, was therefore preferred over a completely custom design. This ensured a sturdy and reliable foundation platform, which has already passed the tests necessary for commercialisation. This choice had the benefit of speeding up development.

The requirement for minimal terrain impact focused the search for a base vehicle into lawnmowers, which are designed for outdoor operations on delicate terrains. The so-called *zero-turn* mowers were of primary interest because of the simplicity of adapting the control interface. These mowers are characterised by two large drive wheels at the rear and two castor wheels at the front. They use differential control of the rear wheels to maneuver without the use of a steering rack; which we would otherwise need to mechanically actuate.

Commercial products were favoured over domestic products given their improved durability and longer run times. The requirement for electric power for the ACS payload demanded an electric powertrain vehicle; this has the added benefit of zero exhaust fumes and low running noise enabling humans to work in close proximity without discomfort.

The platform chosen is an SK-48 Stalker built by Mean Green Mowers.[4] This fulfilled all requirements of size, battery life, and smooth operation on grass. This is shown in Fig. 1.

---

[4] Mean Green Mowers: SK-48 Stalker: https://meangreenproducts.com/sk48-stalker/.

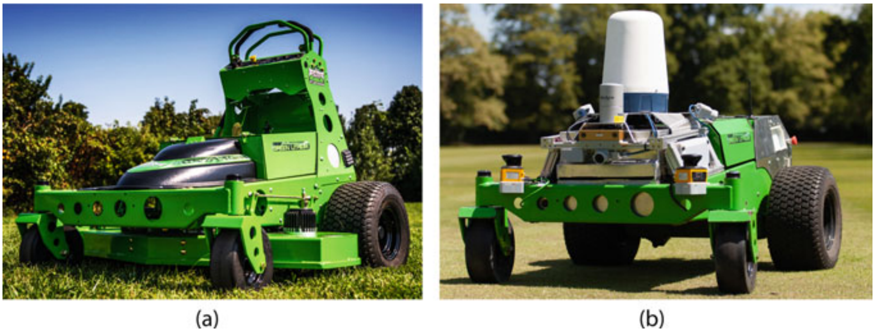


Fig. 1 Photos of the platform before (**a**) and after (**b**) modifications. The modifications are clearly visible with the sensors at the front and the power and control circuitry compartment at the rear

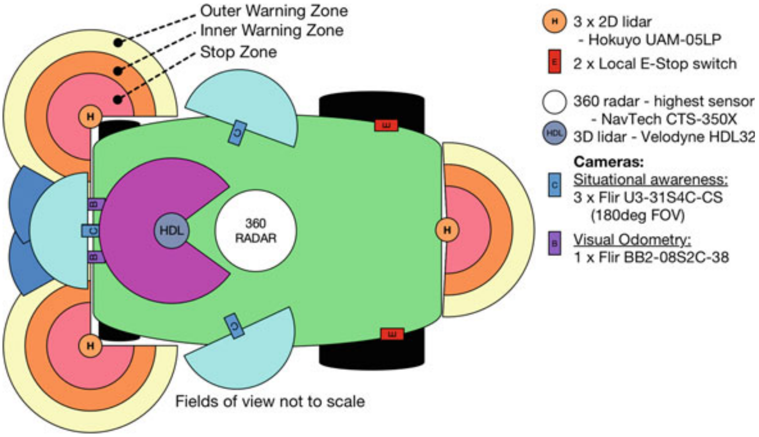

Outer Warning Zone
Inner Warning Zone
Stop Zone
H
HDL
360 RADAR
C
B
E
Fields of view not to scale
3 x 2D lidar - Hokuyo UAM-05LP
2 x Local E-Stop switch
360 radar - highest sensor - NavTech CTS-350X
3D lidar - Velodyne HDL32
Cameras:
Situational awareness:
3 x Flir U3-31S4C-CS (180deg FOV)
Visual Odometry:
1 x Flir BB2-08S2C-38

Fig. 2 The fields of view and arrangement of the primary and SCS sensors

4 Sensor and Computer Payload

The Hulk is equipped with a range of different sensors of heterogeneous modalities. This provides redundancy and resilience in its operation with each complementing the next. Their general layout is shown in Fig. 2.

The main requirement of the platform, as described in Sect. 2, is to be capable of autonomous operation in any weather. For this reason, the main sensor for perception is a CTS350-X Navtech Frequency-Modulated Continuous-Wave (FMCW) scanning radar,[5] which is mounted on top of the platform with an axis of rotation perpendicular to the driving surface. It is characterised by an operating frequency of 76 GHz–77 GHz, yielding up to 2000 range readings with a resolution of 0.175 m, each constituting one of 400–800 azimuth readings—each 2% wide—and a scan rotation rate of 4–8 Hz. Compared to other sensors, like cameras and lidars, the radar suffers far fewer ill effects from deleterious weather conditions such as rain, fog, or snow. This increases the Hulk's resilience and reliability in all conditions. The radar can be used for obstacle detection, odometry and localisation [4, 6].

---

[5] CTS350-X Navtech scanning radar: https://bit.ly/2Ur5yIO.

The platform is also equipped with a number of cameras and a lidar to perceive the world. Mounted on the front of the platform facing towards the direction of motion is a Point Grey Bumblebee2 stereo camera.[6] The camera is characterised by 1032 × 776 × 3 resolution, 20 Hz, 1/3″ Sony ICX204 CCD, global shutter, 3.8 mm lens, 65% HFoV, 12 cm baseline. Its primary tasks are Visual Odometry (VO) [11] and forward obstacle and terrain assessment [5].

Three-Point Grey monocular cameras[7] with fisheye lenses are mounted on the platform. These cameras are characterised by 2048 × 1536 × 3 resolution, 55 Hz, 1/1.8″ Sony IMX265 CCD, global shutter. The main role of the cameras is used for obstacle detection and situational awareness for when the Hulk is being teleoperated.

Further to these sensors, the platform is equipped with a Velodyne HDL-32E Lidar[8] which presents an 80–100 m range, 360% horizontal Field of View (FoV), and 40% vertical FoV. This sensor is primarily used for obstacle detection.

In addition to the aforementioned sensors, the platform is equipped with three safety sensors, which will be covered in more detail in Sect. 5.1. These are Hokuyo UAM-05LP-T301 laser scanners[9] with an FoV of 270%.

Finally, we include in the sensor payload a 3DM-GX4-45 Microstrain inertial sensor[10] with drift gyros. This allows GPS data to be captured and GPS waypoint planning to be used if desired.

Due to the experimental nature of the system and its likely deployment to sites with low connectivity (even potentially when connected to a base station), the decision was made to support as much computing power on-board the Hulk as the mechanical platform would permit. It is equipped with a 2.4 GHz 10-core Xeon CPU and Geforce Titan X GPU. These permit both online and offline Deep Learning algorithms to be implemented on-board [13, 14].

5 Safety Systems

In order to provide sufficient confidence in the safety systems on the platform, the design and implementation were carried out in accordance with the European Machinery directive and associated standards:

- BS EN ISO 12100:2010 [1]—Safety of machinery. General principles for design. Risk assessment and risk reduction

---

[6] Bumblebee2 camera: https://bit.ly/2KmT0h5.

[7] Chameleon cameras: https://bit.ly/2D2Zrjt.

[8] Velodine HDL-32E: https://velodynelidar.com/hdl-32e.html.

[9] Hokuyo UAM-05LP-T301: https://bit.ly/2I8C1gI.

[10] 3DM-GX4-45 Microstrain inertial sensor: https://bit.ly/2Iouh9M.

- BS EN ISO 13850:2015 [3]—Safety of machinery. Emergency stop function. Principles for design
- BS EN ISO 13849-1:2015 [2]—Safety of machinery. Safety-related parts of control systems General principles for design.

The use of a CE marked COTS baseline made this process simpler as only additional sensors and housings required rating. These standards led us through a risk assessment process to determine the required reliability of the system.

The Performance Level (PL) standard defines a methodology for assessing the risk to individuals using or in close proximity to the equipment. This then defines a range of five performance levels that the control system must be built to, from the lowest rating of PL-a through to PL-e. The standard asks three fundamental questions about the risk that the device poses to humans: severity of injury—for Hulk serious, since it is a large and relatively quick vehicle; the frequency of exposure—for Hulk seldom, it is designed to work mostly alone; the ability to avoid injury—for Hulk avoidable as it will be operating outside in large open spaces.

This assessment specifies a PL-c rating for the Hulk which requires an average probability of a dangerous failure per hour to be less than 3×10^6. This informs our design process as will be described in the following sections in which we describe the hardware and software stacks as well as the choice of components for either.

5.1 Hardware Safety-Curtain System

To meet the required PL-c rating we have implemented a Hardware Safety-Curtain System (HW-SCS) on a set of bespoke Printed Circuit Boards (PCBs). It integrates two main safety functions: obstacle detection and ranging (feeding into commands which slow or stop the vehicle), along with an Emergency Stop (E-Stop) function which can be commanded remotely and locally. To achieve the safety rating, each of the three signal sources are PL-c or PL-d rated COTS components or systems:

- Three 2D, 270° FoV lidars, designed for use on forklift trucks and other industrial equipment. These have an essentially uninterrupted FoV around the robot and provide a series of digital I/O lines to communicate discrete signals and commands.
- E-Stop switches[11] mounted on each side of the chassis. These feature two independent switch blocks for redundancy.
- A wireless E-Stop system (Tyro Indus 1S/Gemini).[12]

To achieve the required PL-c rating for safety-stop related functions a Category 3 [2] architecture is used. This requires two independent channels from signal to actuator. Each of the three sources provides pairs of independent outputs, which are connected to parallel circuits within the HW-SCS system. To meet the monitoring

---

[11] E-Stop switches: https://www.schneider-electric.com/en/search/XB4BS8444.

[12] Tyro Indus 1S/Gemini: https://bit.ly/2Z1mC79.

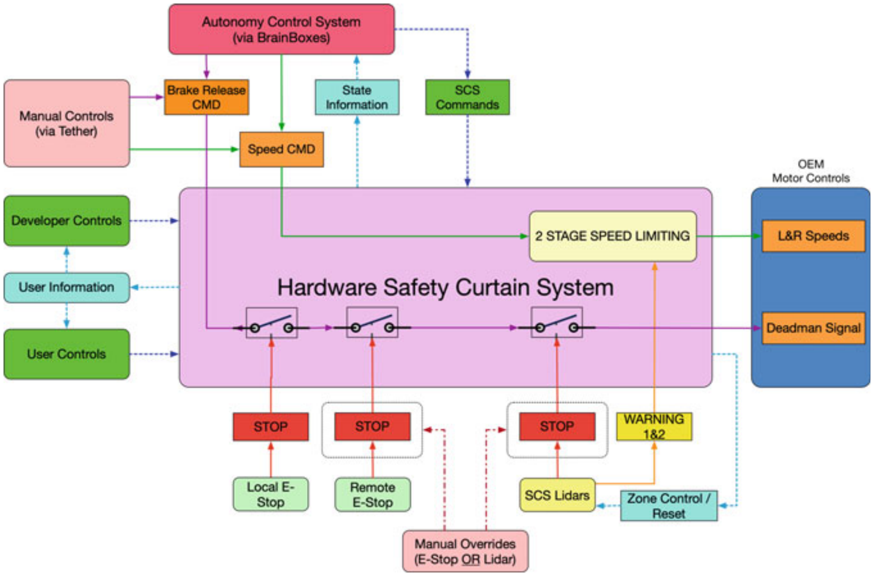

Autonomy Control System (via BrainBoxes)
Manual Controls (via Tether)
Brake Release CMD
State Information
SCS Commands
Speed CMD
OEM Motor Controls
Developer Controls
User Information
User Controls
2 STAGE SPEED LIMITING
Hardware Safety Curtain System
L&R Speeds
Deadman Signal
STOP
STOP
STOP
WARNING 1&2
Local E-Stop
Remote E-Stop
SCS Lidars
Zone Control / Reset
Manual Overrides (E-Stop OR Lidar)

Fig. 3 Simplified block diagram for the HW-SCS

requirement, all control signals are routed through double pole relays. The relays' second channels are used for diagnostic signals which are routed back to the ACS and/or a user interface. Numerous state and health monitoring signals are provided.

To make use of the mower's existing safety features, the HW-SCS system is designed to interface with the Original Equipment Manufacturer (OEM) *dead man* signal in a fail-safe manner. Any break in the signals from the switches or sensors results in a break in the dead man signal, the mower's control system then initiates a Category 2 Stop [3], immediately bringing the vehicle to a controlled stop and applying the motors' brakes, leaving the system powered. Speed signals are ignored by the mower controllers (Fig. 3).

In keeping with the safety standard, when a safety-stop is triggered a separate reset command is required to re-enable control of the vehicle. This prevents the vehicle from restarting the moment the cause of the stop is cleared.

In addition to the safety-critical stop signal, the lidars provide two signals to indicate the presence of obstacles within two *warning* ranges, these are set outside the safety range. Multiple sets of ranges (*zones*) can be pre-programmed into each lidar, selected via discrete digital inputs. The HW-SCS makes use of these warning signals to cap the speed command voltages sent to the Hulk in two stages: affording the ACS and bystanders more time to react and avoid obstacles.

The HW-SCS has been designed to accommodate a range of planned use cases including attaching a trailer, vehicle transport, and emergency recovery. The latter is supported through a tether to connect the OEM controls for manual operation of the mower. For vehicle transport, manual operation, or operation in tight spaces; the safety-stop functions of either the remote E-Stop or the lidars can be temporarily overridden via a keyed switch. The speed limiting function is maintained to assist the operator. A specially modified trailer can be attached through a harness which augments the HW-SCS via lidar(s) on the trailer. When the trailer harness and/or the tether are connected, the SCS automatically switches the lidars to appropriate zone sizes to allow for the presence of a trailer and/or user, as well as navigation in tight spaces.

5.2 Software Safety Curtain

In addition to this hardware level of guaranteed safety, we layer on a Software Safety-Curtain System (SW-SCS). The primary purpose is to gently slow the Hulk down when approaching obstacles so that when the HW-SCS is triggered there is no abrupt and sudden change of speed. This behaviour would be alarming for people near the robot and reduces the acceptance of the robot in their workspace. The software, therefore, sets larger monitoring areas than is done in hardware.

The SW-SCS monitors the range measurements from the same safety lasers used in the hardware safety curtain. When it detects an obstacle it limits the maximum speed command that can be issued. This facilitates a far smoother profile than can be simply and reliably achieved through a hardware-based system.

6 Vehicle Modification

The base vehicle is divided into three main compartments: at the rear sit the motors and associated drive circuity, in the middle lie the batteries and at the fore there is a leaf blower. This leaf blower was removed to provide space for the computer and sensor payload. The manual controls and operator platform at the rear were removed and a new compartment was created for additional power and control circuitry. The battery bay was left untouched and unobstructed for ease of access to the batteries.

6.1 Sensor and Compute Bay

The front compartment houses the greater portion of the ACS, containing the MVC as well as all the primary sensors and some of the power system. The primary sensors are rigidly attached to a chassis which sits over the MVC compartment. This sensor chassis is detachable from the platform chassis, to give easy access to the MVC, whilst maintaining the relative positions of the primary sensors. This defines the outer volume of the ACS compartment and was designed to prioritise the sensor FoVs.

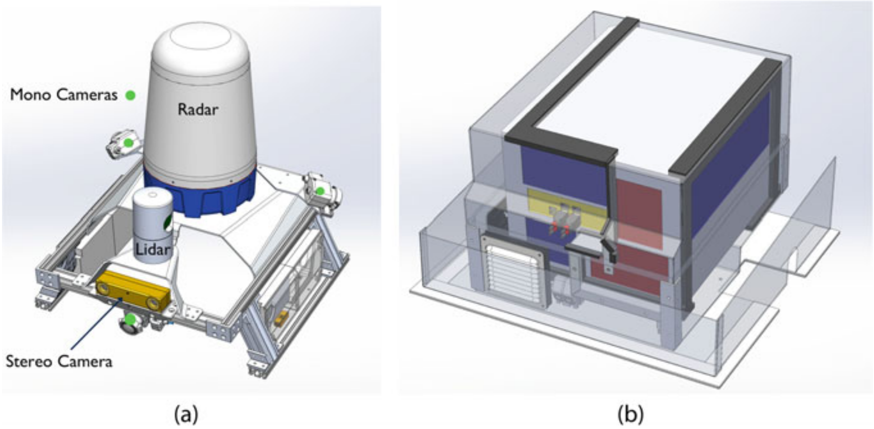


Fig. 4 CAD models of the sensor chassis (**a**) and MVC compartment (**b**). This shows the independent sensor rig formed of Mettalin in **a** and the airflow control in the MVC compartment with inlets in blue and outlets in red

The Computer-Aided Design (CAD) model of the sensor chassis can be seen in Fig. 4a, while the sensor suite is listed and discussed in detail in Sect. 4.

There were a number of requirements for the sensor rig based on the required FoVs.

- The monocular cameras had to cover a full 360%.
- The radar FoV had to be entirely unobstructed.
- The Velodyne FoV should be maximised, and it should see the ground as close as possible to the vehicle.
- The GPS antenna should have an unobstructed view of the sky.
- The BumbleBee 2 needs to be mounted facing forward with a clear view of the ground.

The sensor positions were initially modelled in CAD with their FoVs visualised for fine tuning of their positions. A temporary rig was constructed with the sensors placed as in the CAD model. Field testing was then carried out and data collected for verification of these positions. Adjustments were quick and easy and could be performed in the field. During the design of the MVC enclosure, some sensors did have to be marginally moved from these positions but the changes were approved using the rig and further data collections.

The sensor chassis was built out of Series 5 Metallic and welded plate metal to keep it lightweight and rigid. Four countersunk screws attach it to the platform chassis to allow rapid removal. Removal is kept simple through the use of a single power and data umbilical cord to link it to the ACS compartment. This allows the sensor chassis to be placed next to the platform chassis when removed.

To enable hardware troubleshooting without removing the sensor chassis, two break-out boxes were designed. These are situated on either side of the sensor chassis and provide access to USB and Ethernet ports from sensors and the MVC. In the event of a malfunctioning sensor, these can be used to diagnose whether the fault is likely to be a hardware fault with the sensor itself or a hardware or software fault with the MVC. Only if the MVC is found to be at fault does the sensor rig need removing.

Between the sensor chassis and the cutting deck, a compartment that can fit and cool the computer and GPU was required. The airflow in the compartment is carefully controlled to ensure the correct flow through the multiple air inlets and outlets of the MVC. See Fig. 4b The movement of air is controlled using baffles formed from sheet metal and closed-cell foam. This guides the air through the MVC, over the power transformers and out.

The built-in cooling system of the MVC blows air out so additional fans were added at the inlets to suck cold air in. The required fan specifications were calculated to limit the maximum temperature to 45 °C in an ambient temperature of 18 °C. This requires an airflow of $25 \times 10^{-2}\ \mathrm{m}^{-3}\mathrm{s}^{-1}$. Redundancy is provided through dual fans, each capable of delivering this airflow.

Due to the required airflow, space limitations and ease of access to the enclosure the maximum IP rating attained was IP54. It is intended that a shroud that will sit over the sensor chassis will increase the IP rating but this is yet to be designed. The combination of these three main components allows for a rating of IP55.

6.2 Rear Interface Compartment

At the rear of the Hulk, the redundant manual controls were removed. The rear was slightly lengthened and enclosed to provide a compartment for control circuitry. Two lockable control panels are provided to allow differing levels of access for operators or *expert* users. A subframe and hitch were added to allow towing of a trailer. This was situated to work well with standard garden trailers whilst ensuring it does not impinge on the maximum attack angle the Hulk can achieve. This subframe was created in CAD and analysed using a Finite Element Analysis (FEA) package. This was iterated to ensure a minimum factor of safety of 3.

6.3 Control Interface

The control interface is considerably simplified through the choice of base vehicle. There is no steering rack or steered wheels to mechanically control; the directional control is provided by the differential speed of the two large rear wheels. As built, the speed of these wheels is governed by control sticks that drive two potentiometers. The voltage output of these is the demand signal for each motor controller's torque controller. To convert the Hulk to autonomous operation these manual control lines are intercepted so the ACS can send control signals to the wheels.

OEM quadrature encoders fitted to the wheels are monitored and wheel speeds fed back to the main vehicle computer through an Arduino at 100 Hz. A standard Proportional-Integral-Derivative (PID) controller is then run to provide speed control of the wheels.

Commercial I/O modules[13] are used to output the control signal voltages from the ACS. These are configured to run a *watchdog* which resets the outputs to known safe values if communication with the MVC is lost. In the case of a power event or software failure that causes the MVC to hang these reset after 0.2s bringing the Hulk to an immediate halt.

7 Human–Robot Interaction

HRI is a growing area of interest and research as robots move out of the laboratory. Scholtz [15] defines five distinct interaction categories: Supervisor, Operator, Mechanic, Peer, and Bystander. Roles for which special provision has been included are discussed below.

7.1 Bystander Interaction

The most important role for the Hulk operation is the Bystander role, which includes whomever the platform shares an environment with. The robot will react to the bystander's actions but the goal or intention of the robot cannot be changed by them. Furthermore, the bystander is not expected to have any knowledge of the vehicle, the details of its operation or purpose and may never have seen it before. This means that the bystander actions are completely unpredictable.

We propose three means of seamless interaction with the platform:

- an intuitive and easy to understand display of the Hulk's intentions and knowledge;
- an auditory method of indicating the Hulk's presence and motion;
- the Hulk must demonstrate competence, confidence and purpose in its actions through smooth and deliberate motions.

The first requirement is met through the installation, around the platform, of a ring of LEDs, which indicates that the Hulk has seen a human or obstacle in its SW-SCS. The software safety curtain is run on the MVC and the location of obstacles sent to an Arduino that controls the LEDs. These are lit green unless an obstacle is in the SCS warning zone, when the LEDs that point to the obstacle are lit orange. If the obstacle comes closer to the Hulk and enters the stop zone, the LEDs change to red. This allows the bystander to have confidence when approaching or passing the Hulk that it is aware of their presence and they can carry on safely. If communication with the MVC fails the LEDs will blink red to indicate a fault condition.

---

[13] Brainbox Ethernet I/O module: http://www.brainboxes.com/ethernet-io-module.

The second means of interaction is based on sound. A sounder has been installed on the top of the platform so that the Hulk's approach can be signalled to bystanders, including those who are visually impaired. A critical characteristic for the emitted sound is a balance between adequately warning people of the Hulk's approach, whilst not being too irritating for bystanders.

The third and final requirement is met with two different methods. On one hand, the Hulk's acceleration and jerk are limited to maintain smooth velocity rates, with the added benefit of ensuring softer interaction with the terrain. On the other hand, the SW-SCS enforces a gentle slowing of the robot when approaching an obstacle, which again signals the Hulk has seen and is reacting. Combined these ensure the Hulk responds gradually and gracefully to people in its orbit, whilst the HW-SCS can enforce rapid deceleration when required.

7.2 Supervisor and Operator Interaction

The Hulk will operate independently and autonomously; nevertheless, operator oversight is required for testing and commissioning. This is facilitated through a software application called the Small Robots Command Centre (SRCC), which is run on a remote computer and connects to the robot wirelessly, allowing them to monitor the platform and send command signals. The MVC is equipped with a wireless card and wireless antenna to enable this (Sect. 4).

Figure 5 shows a screenshot of the application in operation. The user interface has been designed with touch screens in mind to facilitate the use of tablet computers in

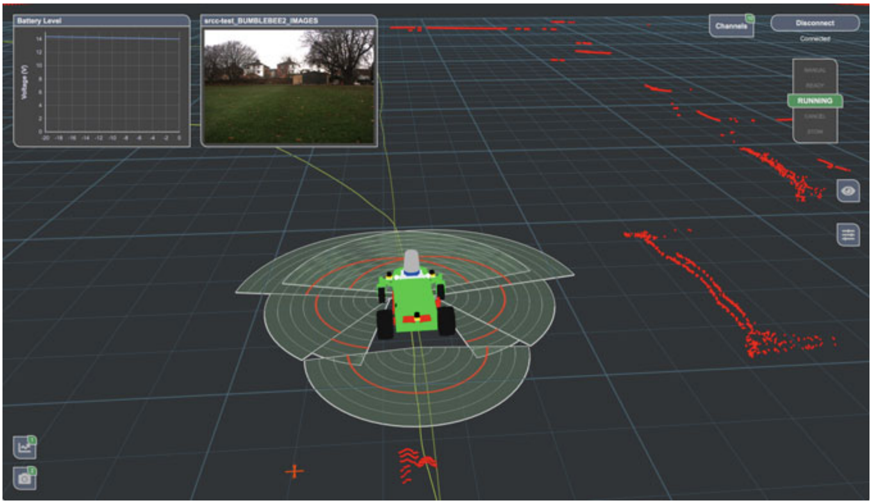


Fig. 5 Screenshot of SRCC remote monitoring software in operation outside at the test site. The safety lasers and SW-SCS state are shown in the 3D workspace and state variables are displayed in overlays. Visualisation elements can be easily turned on and off

the field. The main part of the display is filled with a 3D representation of the robot, its current state, and its immediate environment; including the projected laser scans, radar scans, odometry trace and planned route. The SCS state is shown with projected cones marking out the safety zones in each laser, which share the same color scheme as the HRI LEDs that ring the vehicle. The autonomy state is always clearly visible and other internal robot state variables are easily accessed from a pop-up panel.

7.3 Mechanic Interaction

Although the Hulk will primarily work autonomously when deployed, daily interactions with mechanics are required when the Hulk needs to be charged and data downloaded for analysis. This is the only manual intervention required in a proposed operational deployment of the Hulk, where at all other points the vehicle will be operating autonomously.

To facilitate these tasks, a dedicated light on the top of the platform is connected directly to the HW-SCS, as mentioned in Sect. 5.1. This visually indicates that the Hulk is prevented from moving by the HW-SCS, allowing the mechanic to safely approach and get into close proximity of the robot.

The same visual approach has been used to guide the required steps for connecting the Hulk to power for charging and downloading collected data. Two blue lights indicate that the power supply has been connected—one for batteries and one for the ACS—a green LED indicates that the Ethernet networking is working properly.

8 Conclusions and Future Work

In this paper, we described the design and implementation of a robotic platform for long-term autonomous operation in outdoor environments, impervious to adverse weather and lighting conditions. A further requirement was to allow the robot to operate on delicate surfaces, as the ones in agriculture fields or in manicured lawns. Safety and ease of use were fundamental characteristics of the platform too.

The design started from the choice of a base vehicle, which was chosen to be a *zero-turn* mower, then modified for fly-by-wire operation and equipped with a full sensor suite and computing payload. To enable remote long-term autonomy in diverse environments, several layers of redundant safety systems were designed and installed and the entire assembly was designed to be weatherproof.

In the immediate future, the computing hardware (MVC and GPU) as described in this paper will be used to support vision and radar algorithms requisite for a mapping and localisation system. These will be integrated into the software stack and tested in a kilometre-scale area defined by an extensive set of lawns and managed by a staff of groundskeepers, with whom Hulk will have to interact with.

We may also design a charging station and procedure to achieve completely autonomous charging (perhaps through the use of inductive charging)—capturing the entire operational cycle. From a perspective as described in [7], lifelong mapping and localisation lead to issues around data management and harvesting. These will have to be addressed, as the Hulk is expected to operate continuously, and has a finite storage capability.

References

1. ISO 12100:2010.: International Organization for Standardization: Safety of Machinery—General Principles for Design. Risk Assessment and Risk Reduction. ISO (2010)
2. ISO 13849-1:2015: International Organization for Standardization: Safety of Machinery—Safety-related Parts of Control Systems—Part 1: General Principles for Design. ISO (2015)
3. ISO 13850:2015: International Organization for Standardization: Safety of Machinery—Emergency Stop Function. Principles for design. ISO (2015)
4. Aldera, R., De Martini, D., Gadd, M., Newman, P.: Fast radar motion estimation with a learnt focus of attention using weak supervision. In: IEEE International Conference on Robotics and Automation (ICRA), May 2019
5. Berczi, L.-P., Posner, I., Barfoot, T.D.: Learning to assess terrain from human demonstration using an introspective gaussian-process classifier. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3178–3185. IEEE (2015)
6. Cen, S., Newman, P.: Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018
7. Gadd, M., Newman, P.: The Data Market: Policies for Decentralised Visual Localisation. ArXiv e-prints, Jan 2018
8. Kydd, K., Macrez, S., Pourcel, P., et al.: Autonomous robot for gas and oil sites. In: SPE Offshore Europe Conference and Exhibition. Society of Petroleum Engineers (2015)
9. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000 km: the Oxford robotcar dataset. Int. J. Robot. Res. **36**(1), 3–15 (2017)
10. Nagatani, K., Endo, D., Watanabe, A., Koyanagi, E.: Design and development of explosion-proof tracked vehicle for inspection of offshore oil plant. In: Field and Service Robotics, pp. 531–544. Springer (2018)
11. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry for ground vehicle applications. J. Field Robot. **23**(1), 3–20 (2006)
12. Peters, A.: Safety of the LUTZ pathfinder automated vehicle. In: Proceedings 22nd ITS World Congress, Bordeaux, France. ERTICO (ITS Europe) (2015)
13. Porav, H., Bruls, T., Newman, P.: I can see clearly now: image restoration via de-raining. CoRR, abs/1901.00893 (2019)
14. Porav, H., Maddern, W., Newman, P.: Adversarial training for adverse conditions: robust metric localisation using appearance transfer. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1011–1018. IEEE (2018)
15. Scholtz, J.C.: Human-robot interactions: creating synergistic cyber forces, pp. 177–184 (2002)
16. Simpson, R., Cullip, J., Revell, J.: The cheddar gorge data set. Technical Report, BAE Systems (Operations) Limited, UK (2011)
17. Smith, M., Baldwin, I., Churchill, W., Paul, R., Newman, P.: The new college vision and laser data set. Int. J. Robot. Res. **28**(5), 595–599 (2009)
18. Wise, M., Ferguson, M., King, D., Diehr, E., Dymesich, D.: Fetch & Freight: Standard Platforms for Service Robot Applications (2016)

Effects of Turning Radius on Skid-Steered Wheeled Robot Power Consumption on Loose Soil

Jean-Sebastien Fiset, Meysam Effati, and Krzysztof Skonieczny

1 Introduction

Skid-steered rovers use different velocities for the left and right wheels to maneuver. Due to their mechanical simplicity, maneuverability, and robustness, they are widely used for excavation, construction, military, and planetary exploration applications. Energy-efficient motion is important for any of these applications, especially when planning paths in power-starved environments. However, the power consumption for skid-steered rovers can be high, and also highly variable, compared to other steering mechanisms (such as explicit or Ackermann steering) due to the lateral motion resistance while skidding in a turn. Detailed power modeling is thus essential for this popular class of mobile robots. Although loose soil is frequently encountered in the applications cited above, skid-steered rover power consumption on loose soil has not yet received much attention in the literature.

Preliminary work revealed that the shortest distance path (where the skid-steered wheeled robot performs a point turn, i.e. turn in place using equal and opposite wheel rotations, followed by a straight line and a second point turn) is not the most energy-efficient path on loose soil. This is shown with experimental data collected with two skid-steered wheeled robots showing a significant reduction in energy consumption when replacing the point turn segments with larger radius turns. These field tests were conducted at White Sands National Monument (Fig. 1b) and at the Canadian Space

---

J.-S. Fiset (✉) · M. Effati · K. Skonieczny
Department of Electrical and Computer Engineering, Concordia University,
Montreal, QC, Canada
e-mail: j_fise@encs.concordia.ca

M. Effati
e-mail: m_effati@encs.concordia.ca

K. Skonieczny
e-mail: kskoniec@encs.concordia.ca

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_9

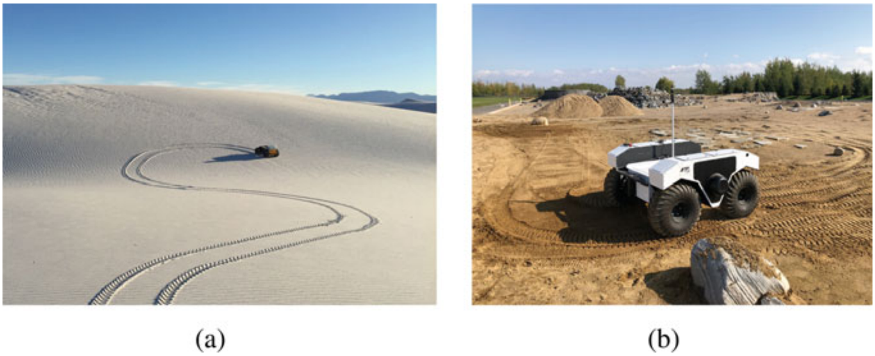

(a)
(b)

Fig. 1 **a** Clearpath Husky A200 on the dunes of the White Sands National Monument. **b** Argo J5 at the Canadian Space Agency Mars analogue terrain

Agency (CSA) Mars analogue terrain (Fig. 1a). Even though this preliminary work focused only on the energy consumption of skid-steered rovers, it further highlights the need for a power model specific to this type of rover on loose soil, with the goal of computing energy-efficient paths.

The state of the art in power modeling for skid-steered rovers is for motion on rigid terrain. One such established class of models is based on instantaneous centers of rotation (ICR) that are estimated online without prior knowledge of terrain properties [1]. The ICR corresponds to the degree of skidding by the rover. Given ICR, the work of [2] (and followed by [3]) uses the kinematic model for tracked mobile robots found in [4, 5] to successfully model power on a rigid terrain. An expanded version of this power model considering terrain elevation was used with a Sample-Based Model Predictive Optimization (SBMPO) scheme to find energy-efficient paths [6] on asphalt and grass. It has also been used by the authors in [7] to optimize the energy consumption of a skid-steered rover on rigid terrain by explicitly considering the radii of the circular arcs along its path.

A recent physics-based power model [8] focuses on friction, the location of the center of mass, the turning radius, and the terrain relief. Their research focused on hard surfaces and the quality of the fit decreased when roving on grass. The work presented in [9] used the dynamic model of skid-steered rovers provided in [10, 11] to model the power consumption.

The contributions of this work are (1) empirically showing shortcomings of existing power models, and discovering a special operating case (wheels dragged against their own rotation) corresponding to where the discrepancy is greatest; (2) isolating significantly higher motor torque and wheel sinkage in controlled experiments replicating the above special case; and (3) linking trends in increased sinkage to higher rover power consumption in promising preliminary modeling.

The remainder of the paper is structured as follows. Section 2 describes the kinematics of skid-steered wheeled robots and a simplified ICR-based power model along with its predictions. Section 3 provides an overview of the equipment used in this research and Sect. 4 shows the results of the tests performed. Finally, Sect. 5 summarizes the conclusions, future work, and practical lessons learned.

2 Power Modeling of Skid-Steered Wheeled Robots

The work of [12] shows that, on hard flat ground, a skid-steered rover's turning radius is inversely related to required torque and thus power consumption. Accordingly, this power modeling analysis explicitly considers the turning radius. This section summarizes existing work on the kinematics of skid-steered wheeled robots and their power modeling when considering slip, on flat ground.

2.1 Kinematic Model

For differential wheeled and tracked vehicles, the location of the center of rotation for the vehicle, ICR_v, is not the same as the center of rotation of each wheel or track set [4], as it can be seen in Fig. 2a. Due to slippage, these instantaneous centers of rotation (ICR) lie outside of the wheels and they are expressed with respect to the vehicle's coordinate frame such that $ICR_l = (x_{ICR_l}, y_{ICR_l})$ and $ICR_r = (x_{ICR_r}, y_{ICR_r})$. As stated by the Kennedy-Aronholdt theorem, the vehicle's center of rotation as well as the two ICRs are located on the same line, parallel to the vehicle X-axis. The

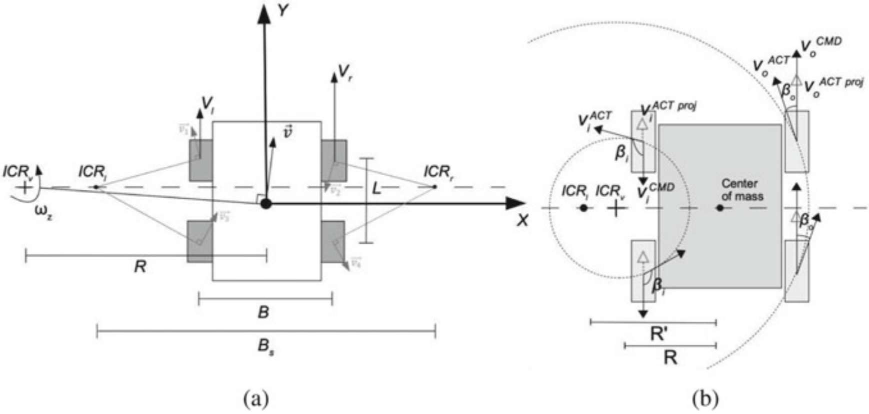


Fig. 2 **a** Schematic figure of a four-wheeled skid-steered robot. **b** Schematic figure of a four-wheeled skid-steered rover performing a turn of radius $B/2 < R < R'$. For the inner wheels, the projection of tangential velocity (or actual velocity of the wheel v^{ACT}) is opposite to the commanded wheel velocity v^{CMD}

distance from the geometric center of the rover to this parallel line is y_{ICR_v} such that $y_{ICR_v} = y_{ICR_l} = y_{ICR_r}$. The position of the left and right instantaneous centers of rotation as a function of the robot's linear velocities (v_x and v_y), angular velocity ω_z, and commanded left and right wheels' velocities (V_l and V_r, respectively; note that these are the control inputs) is reported in the work of [4].

It should be noted that the location of the center of mass has an impact on the location of the left and right instantaneous centers of rotation. As it moves closer to one side of the vehicle, the wheels on this side will slip less, thus resulting in an instantaneous center of rotation closer to those wheels [4].

In this work, the distance between the two centers of rotation is defined as the slip track (B_s).

$$B_s = x_{ICR_r} - x_{ICR_l}. \tag{1}$$

As the ICRs always lie outside of the rover's track due to slippage, the slip track is always greater than the track width, i.e. $B_s \geq B$ [4, 5]. This was validated for both platforms tested (Clearpath Husky and Argo J5), which suggests a generality in the results presented in this work. It is also important to note that, even if R (or x_{ICR_v}) vary within $\pm\infty$, the values of x_{ICR_l}, x_{ICR_r}, and y_{ICR_v} remain within bounded areas for a specific terrain, regardless of the maneuver [4, 5]. Hence, the slip track can be used as a measure of the overall skidding of the wheeled vehicle. A bigger slip track indicates a higher overall skidding, as the amount of wheel velocity contributing to the lateral motion is increased and the amount of wheel velocity contributing to longitudinal displacement is decreased. Accordingly, the ICR positions can help characterize the properties of a terrain [5]. Studies have also shown that knowing the instantaneous center of rotation locations can enhance the pose estimation [1, 4] and be used for power modeling [2, 3, 6].

The following equations are used to express the linear and angular velocities in the body frame when considering slippage.

$$v_y = \frac{V_l x_{ICR_r} - V_r x_{ICR_l}}{B_s} \tag{2}$$

$$v_x = \frac{(V_r - V_l) y_{ICR_v}}{B_s} \tag{3}$$

$$\omega_z = \frac{V_r - V_l}{B_s}. \tag{4}$$

If it is assumed that the left and right instantaneous centers of rotation are symmetric (i.e. $x_{ICR_l} = -x_{ICR_r}$) and that the ICR lies directly on the X-axis, Eqs. 2 and 3 can be simplified to

$$v_y = \frac{V_r + V_l}{2}, \tag{5}$$

$$v_x = 0. \tag{6}$$

If v_y is tangential to the rotation, then it is known that

$$v_y = \omega_z R, \tag{7}$$

where R is the turning radius of the robot, depicted in Fig. 2a. Accordingly, the turning radius R can be defined as a function of V_r, V_l and B_s.

$$R = \frac{B_s}{2}\left(\frac{V_r + V_l}{V_r - V_l}\right). \tag{8}$$

In this paper, R' is defined as the turning radius at which the inner wheels are not commanded to turn. Using Eq. 8, the value of R' for a left turn (i.e. when $V_l = 0$) can be found by the following equation:

$$R' = \frac{B_s}{2}. \tag{9}$$

The first assumption made in this work is that the total absolute velocity v_c of the rover is constant. This total absolute velocity is defined as

$$v_c = \frac{|V_r| + |V_l|}{2}. \tag{10}$$

This assumption is more realistic for skid-steered rovers than a constant forward velocity (v_y) assumption, for example. As turning radius approaches zero, constant forward velocity would require angular velocity, and V_l and V_r, to approach infinity. In real systems, motors producing V_l and V_r saturate at a maximum value. Thus, the stated assumption is more realistic than the one of constant forward velocity.

Note that for $R \geq R'$, V_r and V_l are both positive, and thus the constant v_c assumption is equal to constant forward velocity (Eq. 5). For $R < R'$, on the other hand, V_r and V_l are of the opposite sign, thus the constant v_c assumption is equal to constant angular velocity (Eq. 4). Therefore, the angular velocity of the robot can be rewritten as a function of v_c, using Eqs. 4, 5, 7, and 10.

$$\omega_z = \begin{cases} \frac{2v_c}{B_s} & \text{if } 0 \leq R < R' \\ \frac{v_c}{R} & \text{if } R \geq R' \end{cases} \tag{11}$$

2.1.1 Special Case (when $B/2 < R < R'$)

Further analysis of the skid-steered wheeled robot kinematics, when considering the slip track B_s, led to studying a particular set of turning radii. When the rover is moving along a circular arc path with a radius that is slightly bigger than half of its width ($B/2$) but smaller than R', the inner wheels rotate backward but are actually being dragged forward by the outer wheels. Figure 2b shows the actual and commanded velocity vectors of the inner wheel (v_i^{ACT} and v_i^{CMD}, respectively) when the rover is dealing with this set of turns. If the slip angle β is defined as being the angle between v^{CMD} and v^{ACT}, the inner wheels' slip angle (β_i) in this special case will always be greater than 90°. The presence of a slip angle is specific to skid-steered wheeled robots as the wheels are fixed and lateral movement is required to achieve arc paths.

It should be noted that, going forward, the experiments are done with left-hand turns, such that $V_l = v_i^{CMD}$ and $V_r = v_o^{CMD}$. The power consumption and the inner wheel sinkage during a left-hand turn with a radius in the special case range will be presented and discussed in Sect. 4.

2.2 Power Model

As mentioned in Section 1, there is a lack of power modeling for skid-steering of loose soil; instead, the state of the art is for rigid terrain. On rigid ground, the dynamic friction between the wheels that causes the vehicle to slip during a turn can be modeled using the known ICR-based power model. Based on the work of [2, 3], the power loss of a skid-steered rover can be modeled as

$$P = \mu|\omega_z| \sum_{n=1}^{N} p_n \|\mathbf{a} - \mathbf{C_n}\| + G(|V_r| + |V_l|), \tag{12}$$

where μ is the friction coefficient, ω_z is the angular velocity, p_n is the normal force, $\|\mathbf{a} - \mathbf{C_n}\|$ is the distance between the nth wheel and its respective ICR, and G is the internal rolling resistance coefficient not only due to wheel and soil deformation, but also due to internal resistance from the gear and driving belt.

In this paper, it is assumed that the rover is symmetric and operating on flat ground, causing the normal forces and the distances $d_{ICR} = \|\mathbf{a} - \mathbf{C_n}\|$ to be equal and constant for all wheels. Under these assumptions, and using Eq. 11, the power model can be simplified to

$$P = \begin{cases} \frac{2c_1}{B_s} + c_2 & \text{if } 0 \le R < R' \\ \frac{c_1}{R} + c_2 & \text{if } R \ge R' \end{cases} \tag{13}$$

where c_1 and c_2 are constants and defined as follows:

$$c_1 = \mu v_c m g d_{ICR}, \tag{14}$$

$$c_2 = 2 G v_c, \tag{15}$$

where m is the mass of the rover, g is the gravitational constant, and d_{ICR} is the distance between a wheel and its instantaneous center of rotation.

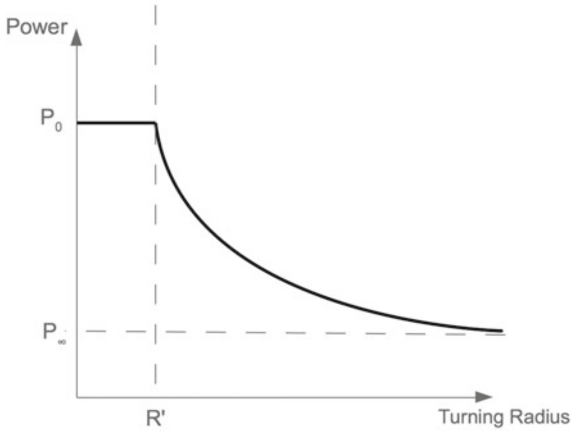


Fig. 3 Predicting power versus turning radius using Eq. 13 on flat ground. Comparison with the experimental results (Fig. 6a) shows the limitation of the approach

As can be seen in Eq. 13 and illustrated in Fig. 3, when operating on flat ground under a constant v_c, it is noted that the modeled power remains constant for turning radii smaller than R'. It will be shown in Sect. 4.1 that this prediction is inconsistent with observed results on loose sandy soil.

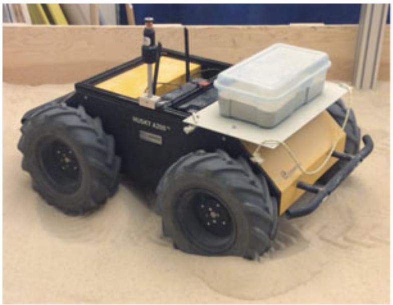
3 Test Equipment and Setup

For the purpose of this research, a popular skid-steered wheeled robot is used to show the limitations of the existing power model on sand. To further analyze the wheel-soil interaction of the inner wheels for the special case identified (i.e. when $B/2 < R < R'$), a single-wheel test bed is used.

It should be noted that, in the laboratory sandbox, the GRC-1 simulant [13] is used to replicate lunar and martian soil. To limit the soil disturbance to improve repeatability, the soil is prepared by loosening it with shovels, compacting it with a tamper, and then leveling it with rakes. This procedure is thoroughly explained in [14]. The sand at the CSA Mars analogue terrain is less characterized but cone penetrometer measurements confirmed that the sand had a significantly higher density.

3.1 Clearpath Husky A200

The platform used to experimentally validate the existing power model is the Clearpath Husky A200. This medium-sized skid-steered wheeled robot is fully supported in ROS. Although this rover has a built-in power measurement system, our preliminary testing unveiled some inaccuracies in the current measurements. An external board using the Texas Instruments INA226 bidirectional current and power



(a)

Parameter	Value
Track Width (B)	0.545 m
Defined Slip Track (B_s)	1.1 m
Length (L)	0.52 m
Wheel radius (r)	0.165 m
Mass (m)	63 kg

(b)

Fig. 4 **a** Clearpath Husky A200 platform mounted with a landmark prism (for position tracking), an IMU and an additional box of sand to place the center of mass of the rover at its centroid. **b** Table of geometric parameters for the Clearpath Husky A200

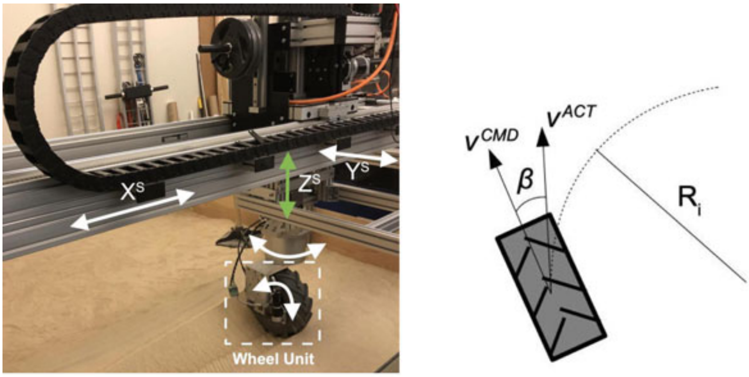
sensor is therefore designed and implemented to measure the current and voltage drawn by the left and right motor controllers and publish the data to a ROS topic. The orientation of the robot is recorded using the VectorNav VN-100R rugged inertial measurement unit (IMU). The left and right wheels' commanded velocities are also recorded from the built-in ROS controller. For the purpose of our testing, an additional weight of 8 kg is placed at the front of the robot to align the center of mass with the geometric center (as shown in Fig. 4a). The equal distribution of the weight (±0.5%) on all four wheels was verified using the Computerscales AccuSet II portable car scale.

The Husky A200 is used at the CSA Mars analogue terrain (Fig. 1b) and the power consumption of the brushed DC motors is measured for a set of turns of radii close to R', with $v_c = 0.3$ m/s. These tests are also repeated in a 2.2×2.2 m sandbox, and the results of these two test campaigns are explained in Sects. 4.1 and 4.2.

3.2 Five-Axis Single-Wheel Test Bed

Figure 5a shows the five-axis single-wheel test bed used for this research. The position of the wheel unit along the X^S and Y^S axes and its orientation around the Z^S axis can be controlled while the wheel unit is free to move along Z^S. The wheel axis is driven by a Maxon RE35 motor and MaxPos 50/5 driver. The load applied to the wheel is set to $1/4$ of the Husky weight by using a counterweight, and the sinkage is measured using a linear potentiometer. The ATI Delta IP60 six-axis force/torque sensor is also mounted between the wheel unit and the wheel itself.

This setup was used to reproduce the circular arc motion of an inner wheel during a set a turns of radii in and around the $B/2 < R < R'$ range. Using the data collected during the Husky tests in the sandbox, the wheel axis was first rotated by β before



(a) (b)

Fig. 5 **a** A Husky wheel placed in the single-wheel test bed. **b** A Husky wheel performing a circular arc path of radius R_i with a commanded linear velocity v^{CMD}, a tangential velocity v^{ACT}, and a slip angle β

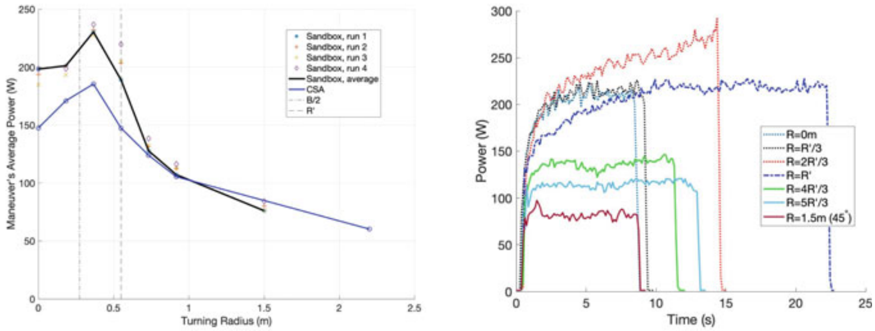
launching the test program. This program used the desired v^{CMD} and v^{ACT} velocities to set the wheel angular velocity and the tangential velocity, respectively. Knowing the rover geometry, the inner wheel's motion radius R_i is computed using the desired turning radius of the rover (R), and the angular velocity around Z^S can be calculated knowing v^{ACT} and R_i. The results of these tests are presented in Sect. 4.3.

4 Results

Experimentally measured power consumption by the Husky on loose soil shows important discrepancies compared to the model for hard terrain presented in Eq. 12. Further analysis of experimental data exposes two possible contributing factors causing the increase in power consumption below R': motor torque and wheel sinkage. A particular focus is put on the inner wheels for the special case identified, i.e. when they are rotating backwards but being dragged forward by the outer wheels.

4.1 Power Model for Hard Ground Limitation on Loose Soil

The purpose of these first tests is to experimentally validate that the power consumption of a skid-steered wheeled robot is not constant below R'. In order to demonstrate this, 90° turns of radii around $R' = 0.55$ m are executed with the Clearpath Husky A200 platform. The average power measured during these maneuvers is depicted in Fig. 6a. In this figure, the results recorded at CSA are compared with the results from



(a) (b)

Fig. 6 **a** Average power consumed during a set of 90° turns performed in the laboratory's sandbox and at the CSA Mars analogue terrain (in green), showing non-constant P for $0 \leq R \leq R'$ (in contrast with Fig. 3). **b** Power consumed over time for a set of turns performed in the controlled laboratory sandbox. For $R \leq R'$ (dotted lines), power rises as the turn progresses

the controlled sandbox tests. It is clear that the power doesn't remain constant for turns where the inner wheels are rotating backwards as we see the power consumption rise significantly (to values 25–30% higher than $P(R')$) and peak in the special case region identified in Sect. 2.1, i.e. when the center of rotation lies between the inner wheels and their ICR ($B/2 < R < R'$). It should be noted that the same trend was observed with two other constant v_c values.

Figure 6b depicts the power consumption with respect to time for the set of turns performed in the GRC-1 soil bed. It should be noted that the total power consumption of the left and right motor drivers is increasing with time in the case of maneuvers with $R \leq R'$. Another important observation is that the time to complete the 90° rotation increases as R approaches R'. This is most likely due to the fact that the rover needs to overcome a higher resistance force, in particular, when the inner wheels are fixed and not excavating the soil (at R'). It should also be noted that similar results were observed when repeating this set of tests at the CSA Mars analogue terrain with another skid-steered wheeled robot, the Argo J5.

4.2 Torque-Proportional Curve

For any electromechanical system, the input electrical power P equals the output mechanical power P_{out} such that $P = \eta P_{out} = \eta \tau \omega$, where η is the power efficiency factor, τ is the output torque and ω is the angular velocity of the wheel in this case. Assuming a constant efficiency, the measurements gathered in the set of 90° turns can be used to draw a torque-proportional curve knowing the following:

$$\tau \propto \frac{P}{\omega}. \tag{16}$$

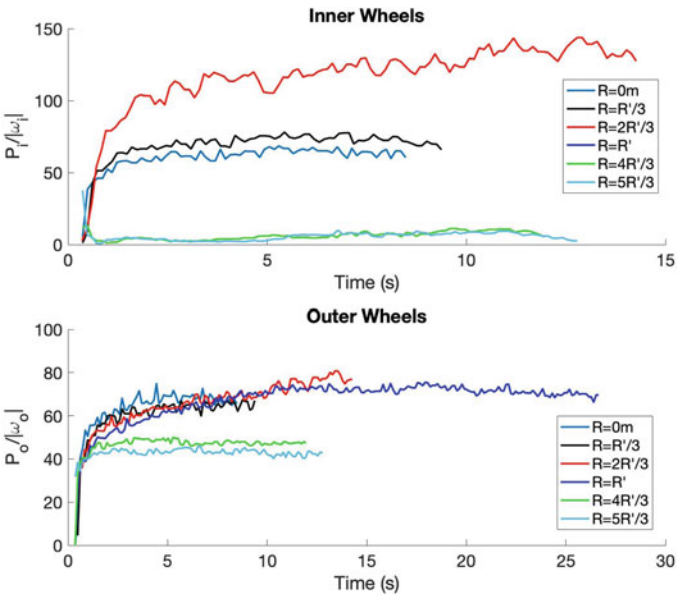


Fig. 7 Inner (top) and outer (bottom) wheels' torque-proportional curve following Eq. 16. For the inner wheels, the $2R'/3$ case significantly needs more torque than the other radii in order to achieve its desired turn

Figure 7 shows the inner and outer wheels' torque-proportional curve using the power measurements and the commanded angular velocity (ω^{CMD}). For the $2R'/3$ case, which corresponds to the turning radius in the special case range presented in Sect. 2.1.1, the inner wheels' torque is clearly isolated as it requires around 70% more torque for these wheels to rotate versus the $R'/3$ case.

The significant increase in measured torque for special case turns, i.e. the $2R'/3$ case, is most likely due to the fact that the grousers need to excavate the sand in front of the forward displacement of the inner wheels. As the wheel progresses and pushes the sand pile, it becomes harder for the grousers to excavate sand from under the wheel to the front, hence requiring more torque to rotate the wheel. A one-dimensional schematic comparison of a normal sand excavation process (when the wheel moves out of the region where sand was excavated) and a scenario when the sand is being compacted in front of the wheel is depicted in Fig. 8. As more torque is required by the motor to rotate the wheel, the system needs to consume an additional amount of electrical power. Therefore, the first argument to explain the peak in power consumption shown in Fig. 6a is identified.

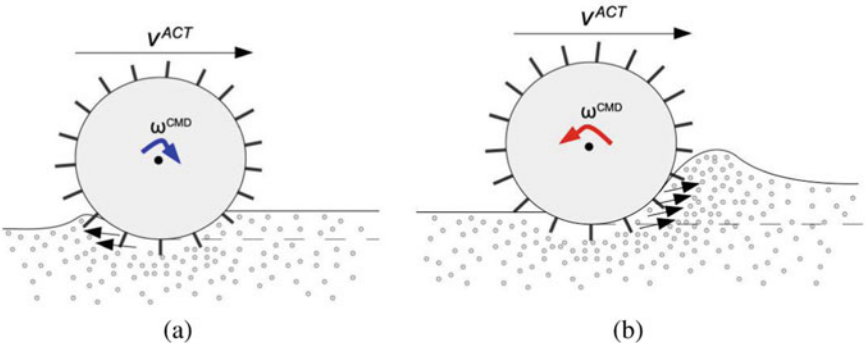


Fig. 8 Schematic comparison of **a** the usual excavation process by the wheel angular rotation (ω^{CMD}) aligned with the wheel progression (v^{ACT}), i.e. when $\beta = 0°$, and **b** the excavation process when the wheel rotation is opposite to the wheel progression, i.e. when $\beta = 180°$

Table 1 Parameters to reproduce Husky's inner wheel trajectories tested

R	R_i m	β	v^{ACT} (m/s)	v^{CMD} (m/s)
$R'/3$	0.141	60°	0.057	0.17
$2R'/3$	0.267	100°	0.042	0.08
R'	0.613	40°	0.05	0
$4R'/3$	0.795	25°	0.127	0.07

4.3 Sinkage

Further tests using the five-axis single-wheel test bed are executed to investigate the sinkage of Husky's inner wheels by reproducing their motion during a turn of radius R. Table 1 presents the inner wheel's turning radius (R_i), slip angle β, tangential velocity (v^{ACT}), and commanded wheel linear velocity (v^{CMD}) used to replicate an inner wheel trajectory for each turning radius R tested.

The sinkage measured by the linear potentiometer for the four test cases is reported in Fig. 9. The first observation that should be made is that the wheel sinks more when $R \leq R'$ than for the $4R'/3$ case, which is consistent with the discrepancies observed between the power model and experimental results as the model doesn't take into account the effects of loose terrain. For skid-steered wheeled robots, maneuvering on loose soil means that the wheels sink as they are slipping and skidding. This requires a higher power consumption in order to push the sand accumulated next to the wheels and, as observed in Sect. 4.1, this increase is more significant below R'.

It should also be noted that for the $2R'/3$ turn, the sinkage is greater than that in the $R'/3$ turn, even if the wheel rotates more than twice faster in the latter case. This additional sinkage means that there is also an additional amount of sand accumulation next to the inner wheels, which results in an increased power required not only to drag the inner wheels but also to bulldoze the sand pile next to the wheels.

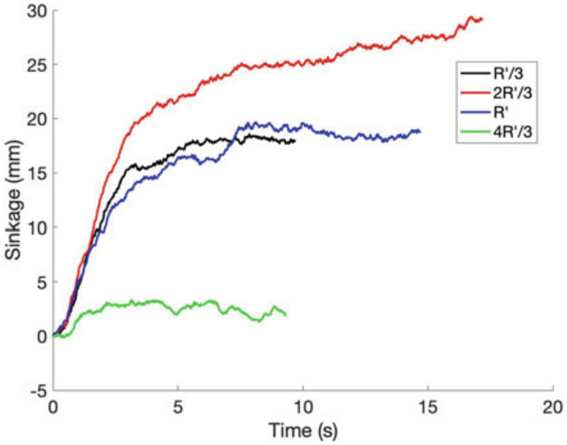


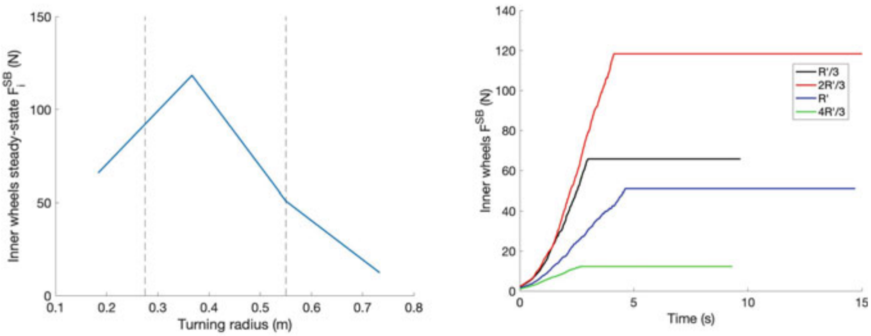
Fig. 9 Measured sinkage versus time for the set of turning radii tested (Table 1). The wheel sinks more in the $2R'/3$ case than in the $R'/3$ even if its rotation speed is less than half

Other measurements recorded during the single-wheel tests yielded no additional insights on the contributing factors increasing the power required to achieve a circular arc path of radii smaller than R'.

4.4 Sand Bulldozing Force (F^{SB})

Wheel sinkage is essential to properly estimate the bulldozing force required to push the sand accumulation next to the wheels. By simplifying the Skonieczny algorithm presented in [15] to model the excavation force throughout a wide-blade tool cut using the GRC-1 soil properties and a few assumptions, the inner wheels' sand bulldozing force can be estimated using the sinkage measurements reported in Sect. 4.3. Figure 10 shows the results of this novel modeling of the inner wheels' bulldozing forces (F_i^{SB}) that the rover needs to overcome in order to achieve its desired turn. The modeled force stabilizes when the sand pile reaches its estimated maximum height, which is a function of the repose angle, slip angle, and steady-state sinkage.

Even as further improvements are currently being applied to the model, these two figures show encouraging results as the steady-state value of the sand bulldozing force (Fig. 10a) depicts the same trend as the power consumption measurements presented in Fig. 6a. The evolution of the force with respect to time for the set of turns tested also replicates the results of Fig. 6b although the modeled force reaches a steady-state value faster than the experimental results. F_i^{SB} is slightly bigger for $R'/3$ compared to R' (even though these two tests recorded similar sinkage) because the slip angle (β) of $R'/3$ is greater than the slip angle of R'. The projection of the wheel perpendicular to the motion increases as β increases, thus resulting in a higher modeled sand pile accumulation and higher bulldozing force.



(a) (b)

Fig. 10 **a** Steady-state value of the inner wheels' sand bulldozing force (F_i^{SB}) versus turning radius. The force peaks at $2R'/3$ like the average power measured shown in Fig. 6a. **b** Sand bulldozing force over time for the set of turns tested

5 Conclusion, Future Work, and Practical Lessons Learned

A key contribution of this work is the identification of a special case in the power consumption of skid-steered wheeled robots. When the turning radius is slightly bigger than half of the width of the rover but smaller than the turning radius where $v_i^{CMD} = 0$, i.e. $B/2 < R < R'$, measured power consumption is significantly higher than existing models predict. As shown in Fig. 6, this special case has an average power consumption approximately 20% higher than for a point turn, and it exhibits the highest rise in instantaneous power with respect to time. Two arguments to explain the increasing power consumption of this special case are studied in this work. First, since the inner wheels' rotation is opposite to their forward progression, i.e. $\beta > 90°$ (as they are dragged by the other wheels, which is intuitively an inefficient way of driving), additional motor torque is required to excavate the sand in a dense area, hence requiring additional power. The second contributing factor comes from the fact that, for this set of turns, the rover experiences more sinkage (in particular, for the inner wheels), and this additional sinkage results in a higher bulldozing force required to push the sand accumulated next to the wheels. Even though the results shown in this paper are specific to the Clearpath Husky platform, the fact that $B_s > B$ for all skid-steered wheeled robots suggests that the trend in the results can be generalized. On the other hand, the magnitude of the results is specific to the platform and soil used.

Future work will model the amount of torque required for the grousers to excavate the sand in a compacted area when $\beta > 90°$. Additional work should also be made in order to predict the sinkage of all of the skid-steered rover's wheels to successfully predict the power required to apply the sand bulldozing force.

Even before a power model is developed to fully account for the contributing factors of increased power listed above, a practical lesson learned in this work is to avoid turns of radii $B/2 < R < R'$ whenever possible when designing an energy-efficient path for skid-steered wheeled vehicles on loose soil.

Acknowledgements The authors acknowledge financial support from Natural Sciences and Engineering Research Council of Canada (NSERC), funding and collaboration from Mission Control Space Services Inc., and thank CSA for facilitating access to their Mars analogue terrain.

References

1. Pentzer, J., Brennan, S., Reichard, K.: Model-based prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation. J. Field Robot. **31**(3), 455–476 (2014)
2. Morales, J., Martinez, J.L., Mandow, A., Garcia-Cerezo, A.J., Pedraza, S.: Power consumption modeling of skid-steer tracked mobile robots on rigid terrain. IEEE Trans. Robot. **25**(5), 1098–1108 (2009)
3. Pentzer, J., Brennan, S., Reichard, K.: On-line estimation of vehicle motion and power model parameters for skid-steer robot energy use prediction. In: American Control Conference (ACC), pp. 2786–2791. Portland, Oregon, USA (2014)
4. Martinez J.L., Mandow, A., Morales, J., Pedraza, S., Garcia-Cerezo, A.J.: Approximating kinematics for tracked mobile robots. Int. J. Robot. Res. **24**(10):867–878 (2005)
5. Reina, G., Galati, R.: Slip-based terrain estimation with a skid-steer vehicle. Vehi. Syst. Dyn. **54**(10), 1384–1404 (2016)
6. Pentzer, J., Reichard, K., Brennan, S.: Energy-based path planning for skid-steer vehicles operating in areas with mixed surface types. In: American Control Conference (ACC), pp. 2110–2115. MA, USA, Boston (2016)
7. Effati, M., Skonieczny, K.: Circular Arc-based optimal path planning for skid-steer rovers. In: IEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec. QC, Canada (2018)
8. Dogru, S., Marques, L.: A physics-based power model for skid-steered wheeled mobile robots. IEEE Trans. Robot. **34**(2), 421–433 (2018)
9. Gupta, N., Ordonez, C., Collins, E.G.: Dynamically feasible, energy efficient motion planning for skid-steered vehicles. Auton. Robots **41**(2), 453–471 (2017)
10. Yu, W., Chuy, O., Collin, E.G., Hollis, P.: Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle. IEEE Trans. Robot. **26**(2), 340–353 (2010)
11. Ordonez, C., Gupta, N., Reese, B., Seegmiller, N., Kelly, A., Collins, E.G.: Learning of skid-steered kinematic and dynamic models for motion planning. Robot. Auton. Syst. **95**, 207–221 (2017)
12. Shamah, B.: Experimental comparison of skid steering versus explicit steering for a wheeled mobile robot, M.S Thesis in Robotics, Carnegie Mellon University (1999)
13. Oravec, H., Asnani, V., Zeng, X.: Design and characterization of GRC-1: A soil for lunar terramechanics testing in Earth-ambient conditions. J. Terramech. **47**(6), 361–377 (2010)
14. Creager, C., Asnani, V., Oravec, H., Woodward, A.: Drawbar Pull (DP) procedures for off-road vehicle testing. In: NASA Technical Report, August, 2017
15. Skonieczny, K.: Modeling the effects of surcharge accumulation on terrestrial and planetary wide-blade soil-tillage tool interactions. Soil & Tillage Res. **176**, 104–111 (2018)

Toward Precise Robotic Grasping by Probabilistic Post-grasp Displacement Estimation

Jialiang (Alan) Zhao, Jacky Liang, and Oliver Kroemer

1 Introduction

Grasping is one of the most fundamental skills for robots performing manipulation tasks. Grasping allows a robot to gain control over objects and subsequently use them to perform specific interactions or use them as tools. Recent work on grasping has largely focused on the problem of getting the object in the hand in some manner. However, many manipulation tasks will require the object to be held in a specific and known pose. For example, to insert a peg in a hole, the robot should apply a firm grasp at the far end of the peg from the insertion point. The robot will also need to have an estimate of the object's pose relative to the hand to perform the actual insertion task. Some grasps will allow the robot to constrain the object more than others, reducing the variance in the object's pose, and thus allow for easier in-hand localization.

In this paper, we address the problem of predicting object displacements during grasping. We propose a method that uses two neural networks—the first predicts whether a grasping action will result in a successful grasp that allows an object to be lifted, and the second predicts a distribution over post-grasp object displacements. The robot then selects grasps with a high success probability and low object displacement variance. The predicted mean object displacement can then be used to

---

J. (Alan) Zhao (✉) · J. Liang · O. Kroemer
Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue,
Pittsburgh, PA 15213, USA
e-mail: alanjz@cmu.edu

J. Liang
e-mail: jackyliang@cmu.edu

O. Kroemer
e-mail: okroemer@cmu.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_10

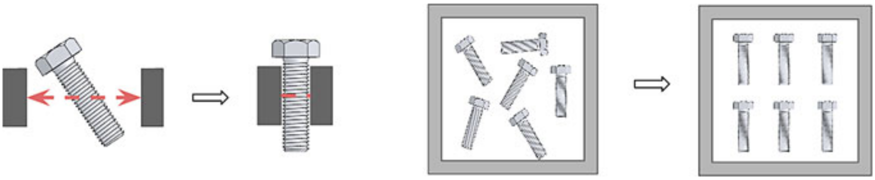


Fig. 1 Example of post-grasp object displacement that our method predicts (left) and palletizing as an example application that requires precise grasp planning (right)

estimate the in-hand pose for downstream tasks such as assembling and palletizing objects (see Fig. 1).

Although our system is trained only in simulation, we were able to successfully deploy the networks on a real Franka Panda robot for precise grasping of 3D printed industrial parts without further fine-tuning. Videos, datasets, and supplementary material are available at: https://precise-grasping.jialiangz.me.

2 Related Works

To generalize grasps between objects, many recent works have used data-driven grasp synthesis [1] techniques to predict high-quality grasps from visual observations. While early works used hand-tuned visual features [21], recent methods have focused on using Convolutional Neural Networks (CNNs) to learn grasp quality functions from a large amount of training data [17, 18, 23].

Robots can collect training data for grasping in a self-supervised manner by attempting thousands of random grasps and observing whether or not they result in successful lifts [15, 16, 19]. By contrast, collecting grasping data in simulation can be much faster and less costly. However, learning on simulation data often suffers from the simulation-to-reality (sim2real) gap, where the visual appearance and object dynamics of simulated data deviate from their real-world counterparts. Methods for overcoming the sim2real gap include domain randomization [22] and domain adaptation [2, 13]. For grasping, the sim2real gap can be reduced by using only depth images for the visual input. Depth-only grasping methods used in [7, 10, 17, 18] do not require further fine-tuning or domain adaptation to perform well in the real world.

Recent works have extended data-driven grasp synthesis for task-oriented grasping, where the system plans grasps that optimize the success of downstream tasks. Detry et al. [7] had human expert label parts of objects that are suitable for tasks like handover and pouring. They trained a CNN to segment depth images based on task suitability to guide the grasp selection. Other works forgo human labels and use simulations and self-supervision to jointly optimize grasp selection and policies for downstream tasks such as sweeping and hammering [9] or tossing [24].

Due to noise in sensing and actuation, as well as unknown object properties, the grasp that the robot intends to execute is often not the grasp that is actually achieved. Jentoft et al. [14] analyzed sources of grasp variations and quantified basins of attraction for multi-fingered grasps. Dogar and Srinivasa [8] learned from human strategy and used pushing to funnel the clutter of objects before grasping to reduce uncertainty. Gupta et al. [11] explicitly learned a noise model to compensate for the actuation noise of low-cost robot arms; while this method improves grasp success, it does not explicitly optimize precise grasps. Chen et al. [4] combine a probabilistic signed distance function representation of object surfaces from depth images with analytical grasp metrics to plan grasps that optimize small post-grasp object displacements.

Instead of choosing precise grasps that minimize object displacement, other works have explored estimating the in-hand object poses using additional sensory signals, e.g., vision and tactile [3]. To address the challenge of in-hand occlusions, Choi et al. [5] trained a CNN to segment out robot grippers such that localization can be done on only the object-relevant pixels, and Izatt et al. [12] used tactile sensing to provide point cloud data on the occluded parts of grasped objects.

In this work, we address the challenges of optimizing grasps for tasks that require precise post-grasp object poses. Given this context, we note that it is acceptable for a grasp to result in a significant object displacement as long as the robot can reliably predict the displacement and adapt the task execution accordingly. Our approach stands in contrast with previous works as we train a CNN to predict the expected post-grasp object displacement and the variance of the displacement. In this manner, the grasp planner can choose grasps that are robust and have the lowest displacement variance.

3 Learning Precise Robotic Grasping

In this section, we describe the problem of precise grasping and explain our approach for addressing this problem.

3.1 Problem Statement

Our proposed approach addresses the problem of estimating a distribution of post-grasp object displacements of top-down, parallel-jaw grasps of singulated objects lying on a flat surface. The method needs to generalize over novel objects unseen during training. We are motivated by the palletizing application and therefore focus our experiments on rigid objects commonly found in industrial and manufacturing settings, such as gears, brackets, and screws.

Let the initial pose of an object on the work surface be $\mathbf{p}$. We assume that during manipulation, the object can only undergo translational movements $(\Delta x, \Delta y, \Delta z)$ and planar rotation $\Delta\theta$, and we define the post-grasp object displacement as $\Delta\mathbf{p} =$

$[\Delta x, \Delta y, \Delta z, \Delta\theta]^T$. As the robot will not be able to use additional sensors to perform in-hand localization, it needs to predict the displacement $\Delta\tilde{\mathbf{p}}$ based on the object pose $\mathbf{p}$ and observation $\mathbf{o}$. The observation $\mathbf{o} \in \mathbb{R}^{64\times 64}$ is a depth image of the object. We use either object-centric full images or grasp-centric image patches of the object for the observations. In both cases, the image size is 64 × 64 pixels.

A grasp $\mathbf{g}$ has 4 degrees of freedom $\mathbf{g} = [g_x, g_y, g_z, g_\theta]^T$. The position parameters $(g_x, g_y, g_z) \in \mathbb{R}^3$ denote the location of the grasp relative to the object's geometric center $\mathbf{p}$. The orientation parameter $g_\theta \in [-\pi, \pi)$ denotes the planar rotation of the gripper about an axis orthogonal to the table surface.

Rather than predicting the post-grasp object displacement $\Delta\mathbf{p}$ directly, our networks instead predict the post-grasp *grasp* displacement $\Delta\mathbf{g}$, i.e., the difference between the grasp parameters and the realized grasp pose relative to the object coordinate frame, which is located at the object's geometric center. This grasp displacement is then converted back to the object displacement $\Delta\mathbf{p}$ for reporting the results.

3.2 Overview of the Approach

The proposed approach consists of three parts: (1) predicting the grasp quality, (2) predicting the distribution of post-grasp displacements, and (3) combining these predictions to choose robust and precise grasps .

Grasp Quality Prediction Following the notation in [9], we define the grasp quality Q of grasp $\mathbf{g}$ with observation $\mathbf{o}$ as the probability of a successful lift S using the grasp $Q(\mathbf{g}, \mathbf{o}) = P(S = 1|\mathbf{g}, \mathbf{o})$. We learn this mapping $Q(\mathbf{g}, \mathbf{o})$ using a neural network that we refer to as the Grasp Quality Network (GQN).

Grasp Displacement Prediction Let $\Delta\mathbf{g}$ denote the distribution of the post-grasp displacement in the object frame. We assume that the displacement follows a Gaussian distribution:

$$\Delta\mathbf{g} \sim \mathcal{N}(\mu(\mathbf{g}, \mathbf{o}), \sigma^2(\mathbf{g}, \mathbf{o}))$$

with mean $\mu(\mathbf{g}, \mathbf{o}) = \{\mu_x, \mu_y, \mu_z, \mu_\theta\}$ and variance $\sigma^2(\mathbf{g}, \mathbf{o}) = \{\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\theta^2\}$. We learn a neural network to predict the mean and variance of $\Delta\mathbf{g}$, which we refer to as the Grasp Displacement Network (GDN). This network allows the robot to reason about the stochasticity of the grasps and select grasps that minimize the variance over the resulting object poses.

Precise Grasp Planning Given the two learned networks, we can form a grasp planner that chooses grasps with a high probability of successfully lifting the object as well as low variance over the resulting object pose.

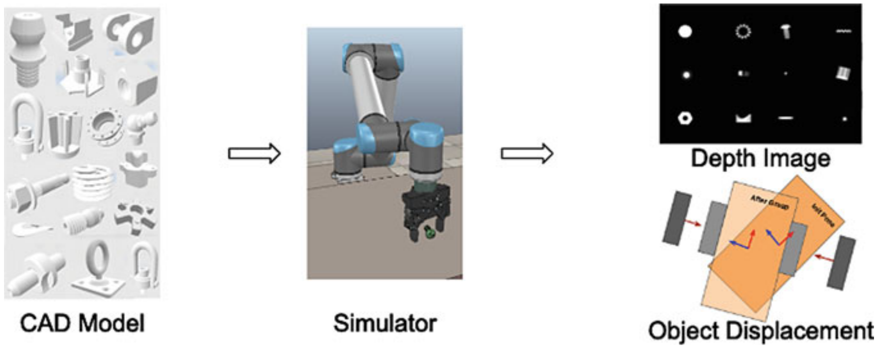


Fig. 2 Simulation Data Collection. We collected a dataset of 1011 industrial parts (left). During simulation, all objects are set to have uniform density and the same coefficient of friction. We uniformly sampled top-down grasps across the object and evaluated whether or not the grasps resulted in successful lifts (middle). In addition to lift success, we also record the top-down depth images that are used as inputs to our grasp quality and grasp displacement networks (top right). We also recorded the post-grasp object displacement (bottom right)

3.3 Simulation Data

To generate grasping data for training the networks and generalizing between different objects, we collected 1011 CAD models of industrial parts such as gears, screws, bolts, and hinges from an online hardware shop.[1] Then, in simulation, 1000 random grasp attempts per object were generated and simulated. We gathered simulation data using the robotic simulation framework V-REP [20] with Bullet ver 2.78[2] as the physics engine. We assume each object has uniform density.

As shown in Fig. 2, a depth image is captured at the start of each grasp attempt. The robot executes a grasp $\mathbf{g}$, where the grasp center (g_x, g_y, g_z) is uniformly sampled from the bounding box of the object, and the grasp orientation g_θ is uniformly sampled from $[-\frac{\pi}{2}, \frac{\pi}{2})$. The bounding box's coordinate frame is always parallel to the camera coordinate frame. The bounding box is calculated as the minimum area that encloses the entire object in the depth image. In contrast to previous works that sample antipodal grasps [17, 18], we use uniformly sampled grasps as using antipodal grasps may introduce a bias in the dataset when estimating post-grasp displacements. For each grasp attempt $\mathbf{g}$, we record the lift success of the grasp S, the overhead depth image $\mathbf{o}$, and the post-grasp object displacement $\Delta\mathbf{p}$.

For training the networks, we collected a simulation dataset with 1.011 million grasps. After removing objects that are either (1) too hard to grasp (random grasp success rate $<5\%$), (2) too easy to grasp (random grasp success rate $>40\%$), (3) too big (longest axis longer than 15 cm), or (4) too small (longest axis smaller than 2

---

[1] McMaster-Carr, https://www.mcmaster.com.

[2] https://pybullet.org/.

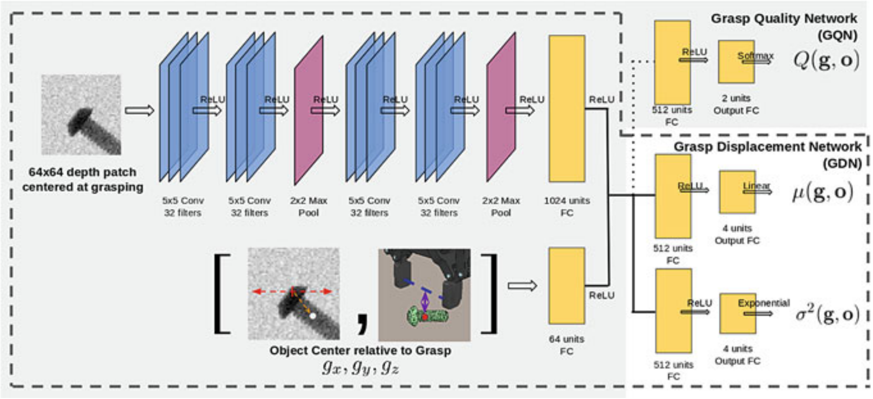

Grasp Quality Network (GQN)
$Q(\mathbf{g},\mathbf{o})$
ReLU
Softmax
512 units FC
2 units Output FC
64x64 depth patch centered at grasping
5x5 Conv 32 filters
5x5 Conv 32 filters
2x2 Max Pool
5x5 Conv 32 filters
5x5 Conv 32 filters
2x2 Max Pool
1024 units FC
Grasp Displacement Network (GDN)
Linear
$\mu(\mathbf{g},\mathbf{o})$
4 units Output FC
512 units FC
Exponential
$\sigma^2(\mathbf{g},\mathbf{o})$
4 units Output FC
512 units FC
64 units FC
Object Center relative to Grasp
g_x, g_y, g_z

Fig. 3 **Grasp Quality Network (GQN) and Grasp Displacement Network (GDN)**. Input to both networks contains a depth image $\mathbf{o}$ cropped at the desired grasp center $\{\mathbf{g_x}, \mathbf{g_y}\}$ aligned to the grasp rotation $\mathbf{g}_\theta$, and the relative translation of the object's geometric center with respect to the grasp center (g_x, g_y, g_z). The GQN predicts grasp quality $Q(\mathbf{g}, \mathbf{o})$, and the GDN jointly predicts the mean $\mu(\mathbf{g}, \mathbf{o})$ and variance $\sigma^2(\mathbf{g}, \mathbf{o})$ of post-grasp displacements. The GQN is trained first, and its learned convolution weights are used to initialize the convolution filters of the GDN. The two networks share the same convolution architecture but not the same weights. We use dropout of 0.5 for the fully connected layers

cm), we have 773 k grasp attempts for 773 objects. Data is split object-wise, with 660 objects used for training and 113 for validation.

3.4 Grasp Quality and Post-grasp Displacement Estimation

We train two types of CNNs—the GQN and the GDN. While the GQN and the GDN share the same convolution architecture, they do not share weights. Rather, the GQN is trained first, and we use its learned convolution filters to initialize the filter weights of the GDN. See Fig. 3 for details.

The **Grasp Quality Network** $Q(\mathbf{g}, \mathbf{o})$ was trained using grasp-centric image patches for the observations, and the translation between the grasp and the object's center (g_x, g_y, g_z). We train the GQN using a binary cross-entropy loss.

The **Grasp Displacement Networks** are trained to predict a Gaussian displacement distribution with mean $\mu(\mathbf{g}, \mathbf{o})$ and variance $\sigma^2(\mathbf{g}, \mathbf{o})$. For the observations $\mathbf{o}$, we have GDN variants that use the object-centric full images or the cropped grasp-centric image patches. The grasp-centric image variants are initialized using the weights from the GQN, while the object-centric image variants are initialized randomly. We also have GDN variants that predict only the mean $\mu(\mathbf{g}, \mathbf{o})$ or both the mean $\mu(\mathbf{g}, \mathbf{o})$ and the variance $\sigma^2(\mathbf{g}, \mathbf{o})$. These four GDN variants are all trained only on successful $S = 1$ grasps and we do not try to predict how failed grasps will displace the objects.

To train the GDN, we form a loss to maximize the log likelihood on the predicted distribution of post-grasp displacement:

$$P(\Delta\mathbf{g}|\mathbf{g},\mathbf{o}) = \frac{\exp\left(-\frac{1}{2}(\Delta\mathbf{g}-\mu(\mathbf{g},\mathbf{o}))^{\top}\Sigma(\mathbf{g},\mathbf{o})^{-1}(\Delta\mathbf{g}-\mu(\mathbf{g},\mathbf{o}))\right)}{\sqrt{(2\pi)^4 det(\Sigma(\mathbf{g},\mathbf{o}))}} \tag{1}$$

$$\mu^*, \sigma^{2^*} = \arg\max_{\mu,\sigma^2}\ \log(P(\Delta\mathbf{g}|\mathbf{g},\mathbf{o})) \tag{2}$$

$$= \arg\min_{\mu_j,\sigma_j^2}\left(\sum_{j=1}^{4}\frac{1}{2}\log(\sigma_j^2) + \frac{(\Delta\mathbf{g}_j-\mu_j)^2}{2\sigma_j^2}\right) \tag{3}$$

where $\Sigma(\mathbf{g},\mathbf{o}) = diag(\sigma^2(\mathbf{g},\mathbf{o}))$. Thus, the loss function to train the GDN is

$$L = \sum_{i=1}^{N}\left(\sum_{j=1}^{4}\log(\sigma^2(\mathbf{g}_j^{(i)},\mathbf{o}_j^{(i)})) + \frac{(\Delta\mathbf{g}_j^{(i)}-\mu(\mathbf{g}_j^{(i)},\mathbf{o}_j^{(i)}))^2}{\sigma^2(\mathbf{g}_j^{(i)},\mathbf{o}_j^{(i)})}\right) \tag{4}$$

The **Grasp Planner** needs to select a grasp based on the learned networks. An ideal grasp for industrial applications needs to satisfy two requirements: (1) the grasp should be stable and (2) the uncertainty over the object's post-grasp pose should be small. We fulfill these two requirements by first running the GQN to select the top **3%** of the scored randomly generated grasps, denoted as $\mathbf{G}$. This step makes sure the planned grasp satisfies requirement (1). We then choose the optimal grasp $\mathbf{g}^*$ as the one that has the lowest displacement variance predicted by the GDN: $\mathbf{g}^* = \arg\min_{\mathbf{g}\in\mathbf{G}}\sigma^2(\mathbf{g},\mathbf{o})$. In other words, among all the top successful grasps $\mathbf{G}$, we select the one with the smallest displacement variance. The expected displacement $\mu(\mathbf{g}^*,\mathbf{o})$ is then used as a correction of the object pose to parameterize downstream tasks such as assembly and palletizing.

4 Experiments

We perform experiments in simulation and the real world to evaluate the performance of both the GQN and the GDNs. We also evaluate how selecting low-variance grasps affects the performance. The GQN and GDN networks are always trained with simulation data only.

4.1 Evaluated Grasp Displacement Models

We compare 5 models for post-grasp displacement estimation:

- **LOWESS**—Locally Weighted Regression.

- **OCFI-M**—GDN with Object-Centric Full Image Input that predicts only the mean grasp displacement.
- **OCFI-M+V**—GDN with Object-Centric Full Image Input that predicts both the mean and the variance of grasp displacement.
- **GCIP-M**—GDN with Grasp-Centric Image Patches that predicts only the mean grasp displacement.
- **GCIP-M+V**—GDN with Grasp-Centric Image Patches that predicts both the mean and the variance of grasp displacement.

LOWESS [6] is a non-parametric regression method that is similar to nearest neighbors except the weight of the neighbors is computed from a Gaussian kernel. Because this model doesn't generalize to novel objects, we only use **LOWESS** as a baseline to predict grasp displacements for *known* objects. Given N grasps on the known training object, and a new query grasp $\mathbf{g}_j$, the predicted grasp displacement is computed by

$$\Delta\hat{\mathbf{g}_j} = \frac{\sum_i^N w(\mathbf{g}_i, \mathbf{g}_j)\Delta\mathbf{g}_i}{\sum_i^N w(\mathbf{g}_i, \mathbf{g}_j)}$$

where $w(\mathbf{g}_i, \mathbf{g}_j) = \mathcal{N}(\mathbf{g}_j|\mathbf{g}_i, \Sigma)$ is the probability density function of the isotropic multivariate Gaussian distribution with mean $\mathbf{g}_i$ and variance Σ evaluated at $\mathbf{g}_j$. We choose the variance terms to be $\Sigma = diag([0.02, 0.02, 0.05, 1.00])$.

The full image GDN variants take as input the uncropped image centered around the object geometric center, instead of the grasp center as is the case with image patches. These models help us understand how important it is for the GDN to focus on local features around the grasp versus global features that describe overall object geometry. Although the GDN variants without variance prediction cannot be used to select grasps by variance, we evaluate against them to see whether or not training to predict this variance helps improve the prediction accuracy of the mean displacements.

4.2 Training GQN and GDN

All the CNN models we used have the same convolutional layer structure, the only differences are the input action sizes and output layer activations. Before training, all action network inputs are normalized to range $[-1, 1]$, and all depth images are reshaped to 64×64. To simulate noise from real depth cameras, we add uncorrelated pixel-wise Gaussian noise of zero mean and 3mm standard deviation to the depth image. We preprocess all depth images by subtracting their mean and dividing by their standard deviation.

We trained the GQN with balanced positive and negative data for 100 epochs with the RMSProp optimizer using a learning rate of 10^{-5} and decay of 10^{-6}. The final accuracy of the GQN is 86.7% on the training set and 85.3% on the validation set.

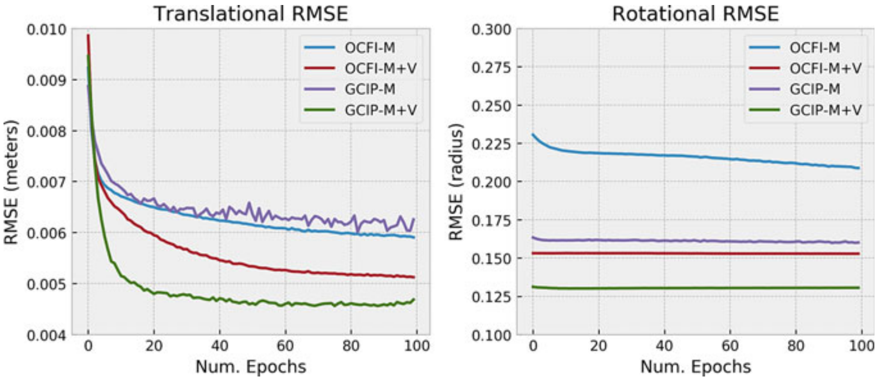


Fig. 4 Validation RMSE of Mean Post-Grasp Object Displacement Predictions for Variants of GDN. We observe that models that incorporate displacement variance prediction consistently outperform ones that do not, and GDN on image patches outperforms operating on the full image

We trained all four variants of GDN in a similar fashion. Root mean square error (RMSE) between predicted mean displacement and actual displacement of each model on the validation set is shown in Fig. 4.

We observe that including the displacement variance prediction improves the RMSE for the predicted displacement means. This improvement is because, with variance prediction, the network is allowed to increase the variance term on data that have high variance instead of fitting a mean which may incur a high loss.

4.3 Evaluation in Simulation

To evaluate the performance of GQN for planning robust grasps, we formed a grasp planning policy that uniformly samples 3200 grasps across the object and picks the grasp with the highest predicted quality. We ran this policy on a set of 130 novel objects not found in the training or the validation set, and performed 30 grasp trials for each object. In simulation, the trained GQN achieved a grasp success rate of **94**.**9**%.

To evaluate the performance of the GDNs, we ran two sets of experiments using different grasp selection policies: (1) select grasps with high predicted qualities according to the GQN and (2) select grasps that have both high quality and low variance. The latter is only applicable for **LOWESS**, **OCFI-M+V**, and **GCIP-M+V** which predict the variances.

All of the models in both experiments were evaluated on a set of 85 objects, of which 50 are from the training dataset, and 35 are from the validation dataset. We perform 35 grasping trials for each object, and only successful grasps were used to evaluate the performance of displacement prediction models.

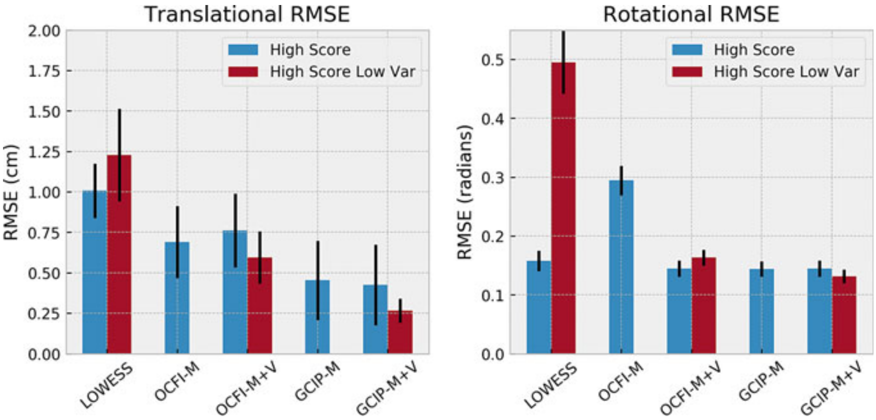


Fig. 5 Translational and Rotational RMSE of Grasp Displacement Predictions in Simulation. Blue bars show results of GDN estimated displacement for the grasps with the highest qualities predicted by the GQN. Red bars show results of choosing grasps that have both high quality and low variance as predicted by a GDN. Grasp displacement models that do not have variance output are not reported for the second set of experiments

Results are shown in Fig. 5. For the high-quality grasps, experiment (1), the average translational error and rotational error are 0.43 cm and 8.29° with **GCIP-M+V**. For the high-quality low-variance grasps, experiment (2), the errors are further reduced to 0.24 cm and 7.01°, respectively. **GCIP-M+V** has the best performance in both experiments. We also observe that choosing high-quality, low-variance grasps generally improves the post-grasp displacement prediction accuracy.

We observe that in both experiments, the proposed **GCIP-M+V** GDN has a good performance in terms of translational displacement estimation, but it does not reduce the error for rotation in comparison to other baselines. This might be because predicting translational displacement is easier than rotational displacement, as the latter needs more information regarding the object's overall geometry.

4.4 Evaluation with Real-World Robot

We use a 7-DoF Franka Emika Panda robot arm for grasping and a Kinect v2 time-of-flight sensor for depth sensing. For our grasping experiments, we 3D-printed 7 novel objects from our dataset of industrial parts that were not used during training. At the start of each grasp experiment trial, we place one object in a 24 cm by 24 cm square region in a bin in front of the robot and directly below the depth camera. Then, the human operator puts a cardboard box over the object before shaking the box for a few seconds. This helps to reduce human bias and randomize the location and orientation of the object.

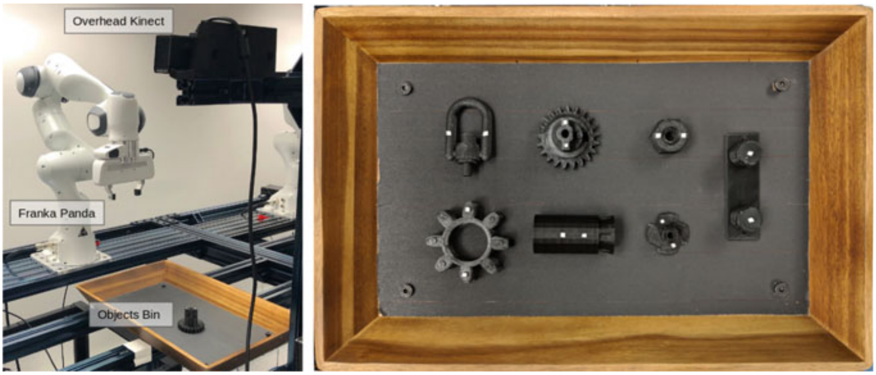


Fig. 6 **Real-World Robot Setup**. We use a 7-DoF Franka Panda robot arm and a Kinect v2 RGB-D camera (left) and 3D-printed test objects used for real-world experiments (right). Two white markers are placed on each object to estimate the post-grasp object displacements

During each trial, we use the GQN trained with simulation data to predict the qualities of randomly sampled grasps from the depth image. A grasp is considered a success if the object is still grasped by the robot after the lift. If a grasp is successful, the robot proceeds to lowering the end-effector back to the grasp pose before releasing the grippers.

By placing the object at the same gripper pose as the grasping pose, we can compute the relative pre-grasp and post-grasp translation and rotations to estimate object displacement. One way to do this is via object pose registration with their known 3D CAD models, but this approach is not robust due to rotational symmetries and the low-resolution of the depth sensor. Instead, we opted for a marker-based approach by making the assumption that object displacements during real robot experiments only occur in a plane. This is done by placing two small, square pieces of white masking tape on top of the object such that the line that connects them crosses the object's geometric center. Their relative translation and rotation after grasping can be robustly determined from registered color images. See Fig. 6 for robot setup and an illustration of our objects with these markers.

4.4.1 Real-World Experiment Results

We carried out the same two displacement estimation experiments with a real-world robot: (1) evaluate high-quality grasps and (2) evaluate high-quality and low-variance grasps. Because we do not have a set of successful training grasps for these objects, **LOWESS** could not be applied. Results are shown in Fig. 7. In the first experiment, by choosing the highest GQN scored grasp in each trial, the robot achieved an **86.3%** lifting success rate.

The **GCIP-M+V** GDN has the best displacement estimation in both experiments. In the first experiment, it achieved a translational RMSE of **0.72 cm** and rotational

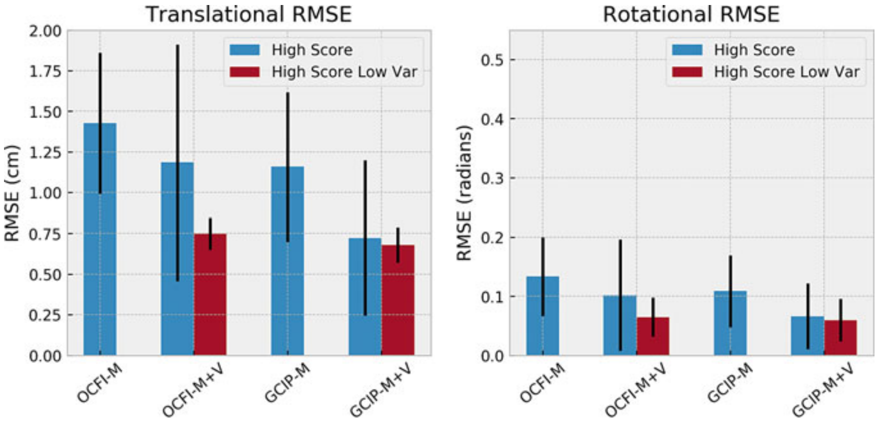


Fig. 7 Real-World Translational and Rotational RMSE of Grasp Displacement Predictions

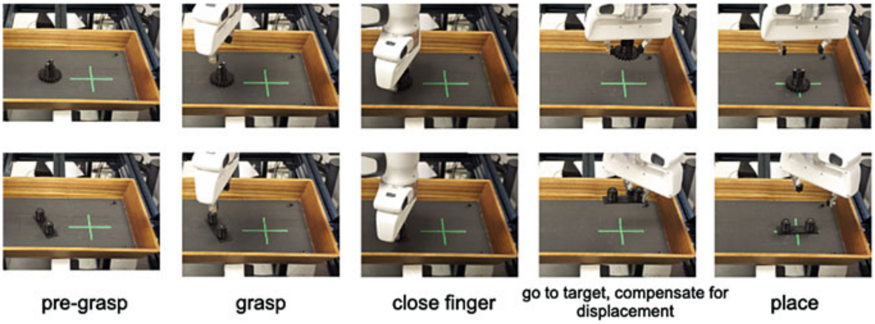


Fig. 8 Example Palletizing Application. We use the post-grasp displacement prediction from our **GCIP-M+V** model to accurately place test objects at a target pose. The target location is marked by the center of the green cross and the target orientation is the horizontal axis of the green cross. By compensating the placing target pose with the estimated post-grasp displacement, the objects can be placed more accurately

RMSE of **3.79°**. These errors are further reduced to **0.68 cm** and **3.42°**, respectively, by choosing high-quality grasps with low predicted displacement variance. Although the mean RMSE values are similar across the two experiments, the error bars are greatly decreased when using the high-score low-variance grasps, which indicates that these models are more consistent and robust across different objects.

In Fig. 7, we show an instance of the palletizing application, wherein we use **GCIP-M+V** to estimate the post-grasp displacement and compensate for the placing action accordingly (Fig. 8).

5 Conclusion

In this work, we propose a method to plan robust and precise grasps by training two networks—one to predict grasp robustness and the other to predict the distribution of post-grasp object displacements. We trained the networks in simulation and deployed them in the real world without further fine-tuning. Experiments in simulation and the real world show our method can effectively predict post-grasp displacements, and choosing grasps that have low predicted variance results in lower displacement prediction errors.

Acknowledgements This project was funded and supported by Epson. This project is also in part supported by National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1745016. The authors also thank Kevin Zhang for his help with real-world robot experiments.

References

1. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis a survey. IEEE Trans. Robot. **30**(2), 289–309 (2014)
2. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (2017)
3. Chebotar, Y., Kroemer, O., Peters, J.: Learning robot tactile sensing for object manipulation. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3368–3375. IEEE (2014)
4. Chen, D., Dietrich, V., Liu, Z., Von Wichert, G.: A probabilistic framework for uncertainty-aware high-accuracy precision grasping of unknown objects. J. Intell. Robot. Syst. **90**(1–2), 19–43 (2018)
5. Choi, C., Del Preto, J., Rus, D.: Using vision for pre- and post-grasping object localization for soft hands. In: International Symposium on Experimental Robotics, pp. 601–612. Springer (2016)
6. Cleveland, W.S.: Robust locally weighted regression and smoothing scatterplots. J. Am. Statist. Assoc. **74**(368), 829–836 (1979)
7. Detry, R., Papon, J., Matthies, L.: Task-oriented grasping with semantic and geometric scene understanding. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 3266–3273 (2017)
8. Dogar, M., Srinivasa, S.: A framework for push-grasping in clutter. Robot. Sci. Syst. **VII**(1) (2011)
9. Fang, K., Zhu, Y., Garg, A., Kurenkov, A., Mehta, V., Fei-Fei, L., Savarese, S.: Learning task-oriented grasping for tool manipulation from simulated self-supervision. Robot. Sci. Syst. (RSS) (2018)
10. Gualtieri, M., Platt, R.: Learning 6-DOF grasping and pick-place using attention focus. Conf. Robot. Learn. (CoRL) (2018)
11. Gupta, A., Murali, A., Gandhi, D.P., Pinto, L.: Robot learning in homes: improving generalization and reducing dataset bias. In: Advances in Neural Information Processing Systems, pp. 9094–9104 (2018)
12. Izatt, G., Mirano, G., Adelson, E., Tedrake, R.: Tracking objects with point clouds from vision and touch. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 4000–4007. IEEE (2017)

13. James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., Bousmalis, K.: Sim-to-real via sim-to-sim: data-efficient robotic grasping via randomized-to-canonical adaptation networks. arXiv preprint arXiv:181207252 (2018)
14. Jentoft, L.P., Wan, Q., Howe, R.D.: How to think about grasping systems-basis grasps and variation budgets. In: Robotics Research, pp. 359–372. Springer (2018)
15. Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al.: Qt-opt: scalable deep reinforcement learning for vision-based robotic manipulation. Conf. Robot. Learn. (CoRL) (2018)
16. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int. J. Robot. Res. **37**(4–5), 421–436 (2018)
17. Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J.A., Goldberg, K.: Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. Robot. Sci. Syst. (RSS) (2017)
18. Mahler, J., Matl, M., Satish, V., Danielczuk, M., DeRose, B., McKinley, S., Goldberg, K.: Learning ambidextrous robot grasping policies. Sci. Robot. **4**(26) (2019)
19. Pinto, L., Gupta, A.: Supersizing self-supervision: learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 3406–3413. IEEE (2016)
20. Rohmer, E., Singh, S.P., Freese, M.: V-rep: a versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1321–1326. IEEE (2013)
21. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. Int. J. Robot. Re. **27**(2), 157–173 (2008)
22. Tobin, J., Biewald, L., Duan, R., Andrychowicz, M., Handa, A., Kumar, V., McGrew, B., Ray, A., Schneider, J., Welinder, P., et al.: Domain randomization and generative models for robotic grasping. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3482–3489. IEEE (2018)
23. Zeng, A., Song, S., Yu, K.T., Donlon, E., Hogan, F.R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., et al.: Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. IEEE Int. Conf. Robot. Autom. (ICRA) (2018)
24. Zeng, A., Song, S., Lee, J., Rodriguez, A,. Funkhouser, T.: Tossingbot: learning to throw arbitrary objects with residual physics. arXiv preprint arXiv:190311239 (2019)

Development of HYDROVER: A Torque-Controlled Hydraulic Rover

**Sang-Ho Hyon, Yusuke Ida, Kosuke Ueda, Junichi Ishikawa,
and Minoru Hiraoka**

1 Introduction

The aim of this study is to realize a general-purpose mobile platform that can traverse irregular terrains such as agricultural or forest landscapes while carrying manipulators and other tools for heavy-duty operations.

The following features are particularly important for this type of robot:

(F1) Terrain adaptability: Able to adapt to even the extreme irregular surfaces and steep slopes, which are common in agricultural or forest landscapes.
(F2) Base stability and mobility: Providing a stable platform that directly supports the operability of the working tools (such as manipulators) to be mounted on the base, or the comfort of the operator. Safety (fall stability) is also a fundamentally important concern.
(F3) Impact resistance: Able to withstand impact from the ground because of a sudden change of the terrain geometry.
(F4) Durability/Weatherability/Low cost.

Numerous and diverse design methods of these mobile robots satisfy these features, ranging from existing all-terrain vehicles to unmanned planetary exploration rovers [1]. We are investigating the possibility of a four-legged, four-wheeled-drive rover built from low-cost hydraulic components (Fig. 2).

Leg–wheel hybrid mechanisms take advantage of features of a leg [2] (ability to discrete ground contact point selectivity and various operating configurations) and of a wheel (simple, high efficiency, and high speed). Many studies have examined kinematics and position-controlled rovers (e.g., [3–7]).

S.-H. Hyon (✉) · Y. Ida · K. Ueda
Ritsumeikan University, Noji-Higashi 1-1-1, Kusatsu, Shiga 525-8577, Japan
e-mail: sangho@ieee.org

J. Ishikawa · M. Hiraoka
Kubota Corporation, Shikitsuhigashi 1-2-47, Naniwa-ku, Osaka 556-8601, Japan

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_11

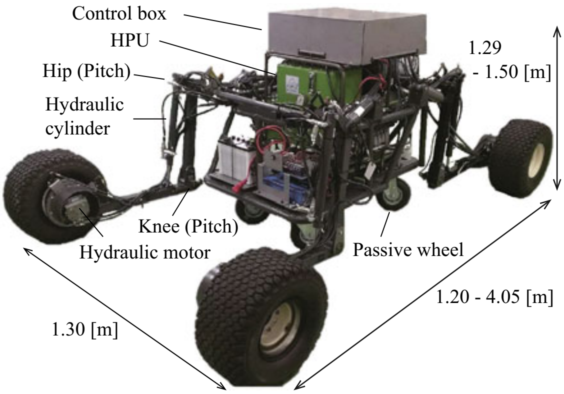

Control box
HPU
Hip (Pitch)
Hydraulic cylinder
1.29 - 1.50 [m]
Knee (Pitch)
Hydraulic motor
Passive wheel
1.20 - 4.05 [m]
1.30 [m]

Fig. 1 HYDROVER-I—The first prototype of a hydraulically driven wheel-on-leg rover

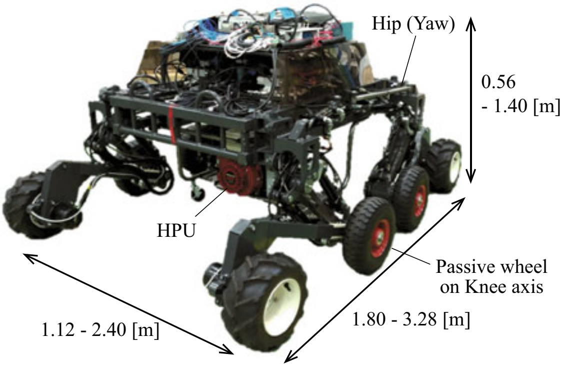

Hip (Yaw)
0.56 - 1.40 [m]
HPU
Passive wheel on Knee axis
1.12 - 2.40 [m]
1.80 - 3.28 [m]

Fig. 2 HYDROVER-II—The second prototype

In this study, we are particularly interested in large-scale *wheel-on-leg rover*, which is useful for heavy-duty tasks for agricultural or forest landscapes. Wilcox et al. [8] developed the Athlete rover (850 kg weight for the earth testing model). This rover has six identical 6-degrees-of-freedom (DoF) limbs that form a hexagon shape. The rover is position-controlled, but the joint torque estimates (from the difference between the motor angle and joint angle) are used to distribute the ground force evenly or resolve undesired internal forces between the limbs. Cordes et al. [9] studied the SherpaTT rover (150 kg weight) having a leg–wheel mechanism with 5 DoF. They showed the performance of positioning the tip of the leg for base attitude stabilization and force leveling. Reid et al. [10] developed the MAMMOTH rover (80 kg weight) to study position-based rough terrain adaptation by way of recursive kinematics algorithms combined with terrain mapping.

Technically, the research most resembling ours is work reported recently by Hutter et al. [11]. They introduced hydraulic torque control to a Walking Excavator (960 kg weight) by Menzi Muck. Thereby, they succeeded in traversing a slope by partially automating the control.

Our project began in 2014; two prototypes of the rovers, shown in Figs. 1 and 2, have been built to explore the problem in the design of the rover. These robots are fully hydraulically powered and made out of low-cost components common in agricultural machines to satisfy the characteristics of (F3) and (F4). In this so-called “wheel-on-leg rover”, the base is supported by four leg mechanisms, and the driving wheel is connected to the end of each leg. Section 2 outlines the robot design focusing on the range of motion, as well as the specification of the hydraulically driven joint.

Unlike many other rovers, this rover can actively control the contact force by directly manipulating the joint torque. Traction control of rovers on irregular terrain is discussed by many researchers (e.g., [12]). We focus on the optimal contact force control method while considering the driving wheels and the auxiliary (passive) wheels. Since the whole body moves compliantly, the robot is expected to be able to accommodate irregular terrain even if it is “blind” (without camera or lidar). It also functions as an active suspension that flexibly absorbs impact from unexpected irregular surfaces. Since the details of the control method are presented in [13], this paper will give an overview in Sect. 3.

Finally, Sect. 4 shows the results of preliminary experiments to investigate force controllability, as well as the results of locomotion experiments on irregular terrain.

2 Robot Design

2.1 Machine Overview

Our hydraulic rover is named *HYDROVER*. The outlook of the first and second prototypes are shown in Figs. 1 and 2. HYDOVER-II surpasses all of the functions of HYDROVER-I.

First, we briefly explain the robot hardware presented in Fig. 2. The basic specifications are shown in Table 1. Its weight is 330 kg, with a base of 1.2 m length, 700 mm width, and standard height at 1000 mm on the nominal configuration. The robot can take various postures, making full use of active/passive DoF.

On each leg, the hip has yaw and pitch axis, the knee has pitch axis, and the driving wheel rotates around the pitch axis that it is attached to at the end of the leg (see Fig. 4). There are a total of 16 active degrees-of-freedom (DoF) on all legs. HYDROVER-II has passive wheels at each knee joint. The wheels are utilized to support the body weight where necessary. In the simulations and experiments shown in this paper, the yaw axis is not used (HYDROVER-I does not have yaw axis at the hip joint, nor passive wheels at the knee).

Table 1 Overall specifications of HYDROVER-II

Mass	330 kg (Leg: 38 kg × 4, Engine: 48 kg)
Size	Base Height: 1000 mm Width: 700 mm Depth: 1200 mm Leg Length: Upper 400 mm, Lower 590 mm Drive wheel Radius: 0.24 mm Passive wheel Radius: 0.14 mm
DoF	16DoF (4DoF × 4)
Actuator	Single rod hydraulic cylinder (Bore: 32 mm) Hydraulic gear motor Proportional valve (10 L/min)
Sensor	Position transducer, Pressure sensor, Encoder, Gyro + Inclinometer (state observer)
Power source	Gasoline engine + hydraulic gear pump

2.2 Actuation and Control System

Figure 3 shows the design of the leg. The joints and wheels are driven by hydraulic cylinders and a hydraulic motor, respectively. Each cylinder at the hip and the knee generates joint torque via a four-bar linkage mechanism (not adopted in HYDROVER-I). The size of the hydraulic cylinders and the link parameters were determined by trial and error considering the kinematics of the robot and some presumed motion pattern. As a result, a single rod cylinder with a piston diameter of 32 mm and a rod diameter of 18 mm was selected. Each component is designed to operate at a supply pressure of 21 MPa, but in the experiment shown in this paper the supply pressure is set to 11 MPa. In this case, the cylinder exhibits a theoretical thrust of 8847 N in the pushing direction and 6048 N in the pulling direction. The position transducer is built in the cylinder, and the joint angle is obtained from the kinematic calculation.

For the driving wheel, a hydraulic gear motor is adopted, which can output a maximum torque of 176 Nm. A simple optical encoder is fabricated and utilized to estimate the rotational speed via the rate of change of the pulse.

A compact, relatively inexpensive proportional valve (maximum flow rate of 10 L/min) was selected as a valve to drive both the cylinder and the motor. The bandwidth of this proportional valve is less than 10 Hz, much lower than industry level servo valves (typically 100 Hz). Nevertheless, the proportional valve is adopted because it is widely available and has the advantage of being almost unaffected by contamination in hydraulic oil. A summary of theoretical joints' output specification

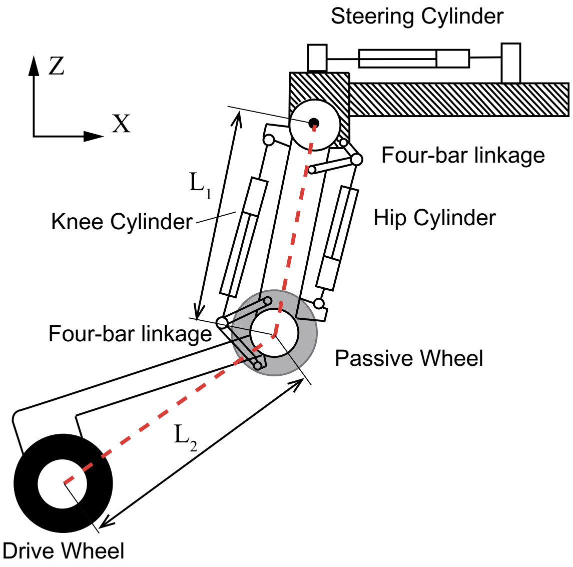


Fig. 3 Leg structure of HYDROVER-II (hind leg)

Table 2 Joint specifications at 11 MPa supply pressure or 10 LPM output flow

	Hip	Knee	Steering	Wheel
Range of motion (°)	−50/70	5/135	−45/45	∞
Max torque (Nm)	460	613	953	113
Max speed (°/s)	398	742	194	153

at a supply pressure of 11 MPa and a maximum valve flow rate at 10 L/min is given in Table 2.

The robot is equipped with a hydraulic power unit (HPU), a real-time controller, and its host PC in addition to the body mechanics. For outdoor experiments, the gasoline engine-driven HPU is mounted on the base. Valves for all of the joints are integrated into manifolds, and the hydraulic power is transmitted to each actuator through hoses. The pressures of each of the chambers of the actuator, and supply and return pressures of the HPU are measured by pressure sensors.

A real-time controller is mounted at the upper center of the base and performs all of the calculations. Gyro sensors and inclinometers are also installed in the same place, and the pitch and roll attitude are estimated via a minimum dimension state observer. The controller sampling rate in the later experiments is set to 1 ms. The host PC is connected to the real-time controller with a LAN cable, which provides a graphical user interface (GUI) and data log to the operator.

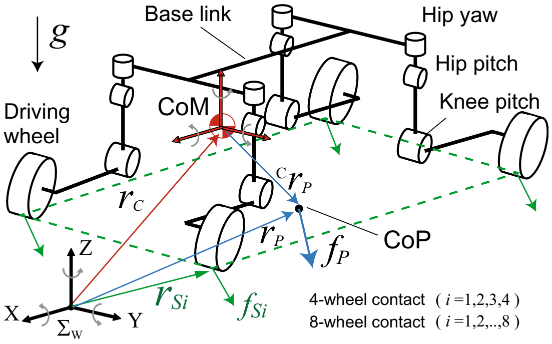

g
Base link
Hip yaw
Hip pitch
Knee pitch
Driving
wheel
CoM
r_C
${}^C r_P$
r_P
CoP
f_P
Z
X
Y
Σ_W
r_{Si}
f_{Si}
4-wheel contact (i =1,2,3,4)
8-wheel contact (i =1,2,..,8)

Fig. 4 Joint configuration and definition of coordinates. $r_C = [x_C,\ y_C,\ z_C]^{\mathrm{T}} \in \mathbb{R}^3$ is the position vector of center of mass (CoM). Each ground contact point r_{Si} is assigned a contact force f_{Si}, which is determined using a desired ground applied force (GAF) f_P. The center of pressure (CoP) r_P invariably lies within the supporting convex hull of r_{Si}

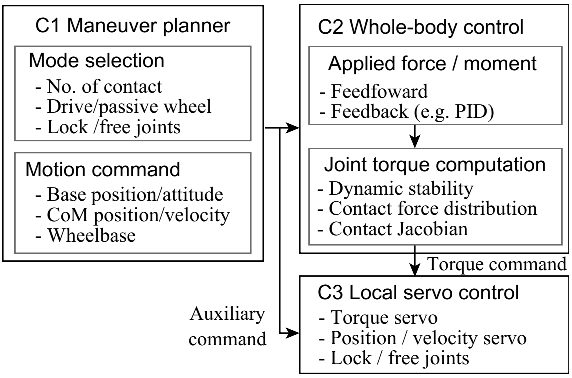

C1 Maneuver planner
Mode selection
- No. of contact
- Drive/passive wheel
- Lock /free joints
Motion command
- Base position/attitude
- CoM position/velocity
- Wheelbase
C2 Whole-body control
Applied force / moment
- Feedfoward
- Feedback (e.g. PID)
Joint torque computation
- Dynamic stability
- Contact force distribution
- Contact Jacobian
Torque command
C3 Local servo control
- Torque servo
- Position / velocity servo
- Lock / free joints
Auxiliary
command

Fig. 5 Control framework for torque-controlled rover

3 Controller Overview

We consider an eight-wheel (including the passive wheels), four-legged robot model and coordinates defined in Fig. 4. However, the control method is applicable to any leg-wheeled robot.

The overall structure of the controller is shown in the block diagram of Fig. 5. It consists of three control modules: C1, maneuver planning; C2, whole-body motion control; and C3, local joint servo.

C1 determines which wheel should be in contact with the ground, based on the measured ground information. It also determines whether the joints are to be locked or not. This setting corresponds to selecting "driving mode" for the robot. For example, on flat ground, it is more energy-efficient to transverse only with the driving wheels similar to a car while the other joints are locked. C1 also commands the traveling speed and postures of the robot, which dictates the position and velocity of the center of mass (CoM) ($\overline{r}_C$, $\dot{\overline{r}}_C$), and the target attitude of the robot's base $\overline{\phi}(t)$.

C2 computes the joint torque necessary to follow the target of the CoM and the base attitude given by C1, while ensuring the terrain adaptability property, similarly to an active suspension. Specifically, the contact forces f_S applied to the environment from each wheel are determined optimally. This problem includes the dynamic stabilization of CoM and base posture. This work employs a passivity-based whole-body motion control algorithm proposed in [14, 15], which is outlined as follows:

(1) Calculate the desired GAF $\overline{f}_P$ and desired CoP.
(2) For given active contact points r_{Si} $(i \in \{1, 2, \ldots, \alpha\})$ and active joints q_j $(j \in \{1, 2, \ldots, n\})$, construct the contact Jacobian ${}^C J_S$ from the CoM to the supporting contact points.
(3) Compose desired forces to control the wheelbase and the base orientation.
(4) Compute the desired contact forces $\overline{f}_S$.
(5) Transform the contact forces $\overline{f}_S$ to torque commands τ for the active joints by

$$\tau = {}^C J_S(\phi, q)^{\mathrm{T}} \overline{f}_S + D\dot{q}. \qquad (1)$$

This algorithm is modified for the application to the rover to accommodate the rolling contact as well as the passive wheels. The terrain information (local inclination of the ground at each contact point) must be incorporated as well. The details are presented in [13].

C3 is a joint servo system that makes the joint torque/velocity/angle follow the target value with minimum error.

4 Preliminary Experiments

4.1 Torque Control of Legs

Since the algorithm given in Sect. 3 presumes the joint torque controllability, the overall performance is strongly affected by it. This section shows the current performance of the torque servo of HYDROVER-II. A simple PID controller with pressure feedback from both of the cylinder's chambers is adopted. The commanded and actual cylinder forces are computed from the corresponding joint torque via the kinematics of the four-bar linkage mechanism depicted in Fig. 3.

Figure 6 shows an example of a gravity compensation experiment of the hind-right leg, where hand-tuned PID gains are set to 2.0/4.5/1.0 for the hip, and 1.6/4.5/0.02 for the knee, respectively. The vibration seen in the graph comes from the internal pressure oscillating, and this does not appear to the force because the amplitude is below the static friction of the piston seals. Finally, a steady-state error of less than 5 Nm is achieved.

We also implemented torque control for the hydraulic motors for driving wheels. Unfortunately, however, we could not obtain good results so far.

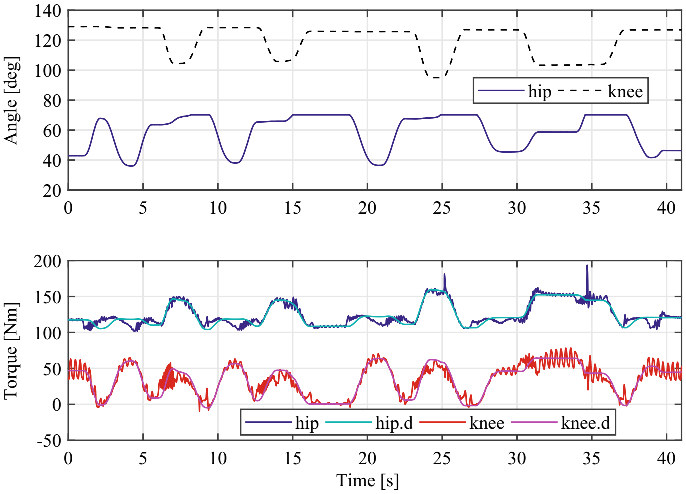


Fig. 6 Gravity compensation experiment on HYDROVER-II rear-right leg. The index ".d" in the legend means the commanded value

4.2 Slope Experiments

We conducted some experiments on the rover using a simple slope. Since the performance of the torque control for the driving wheels is not good, we introduced "local" speed control. That is, each driving wheel is commanded the torque (sim) or valve current (exp) as the high-gain proportional velocity feedback:

$$cmd_i = -K(v_i - \dot{x}_C) \tag{2}$$

where the each velocity v_i is estimated from the wheel speed meters "individually".

Then, we conducted the same locomotion task for the simulation and experiment; then we compared the results. The overall behavior is shown in Fig. 7. The desired locomotion speed is set as $\dot{x}_C(t) = 0.2$ m/s for $t \geq 0.5$ s. The desired CoM height is set to 0.8 m with the proportional feedback gain of 15,000 N/m. The attitude feedback gains are set to 10 Nm/° for both pitch and roll. The joint damping is set only in the simulation: 3 Nms/°

The time evolution of the principal states is depicted in Fig. 8. The horizontal position of CoM is estimated from the averaged speed of the four driving wheels. The height is well regulated to the desired value with a maximum error of 0.08 m. Errors in the base attitude are regulated to within 5° in the pitch axis, and to 8° on the roll axis. No significant difference is found between the experiment and simulation, except for the joint torque (bottom panels). A possible reason is a large friction in the low-cost components (joints and actuators). The difference could be mitigated if we introduce the hydraulic model into the simulator.

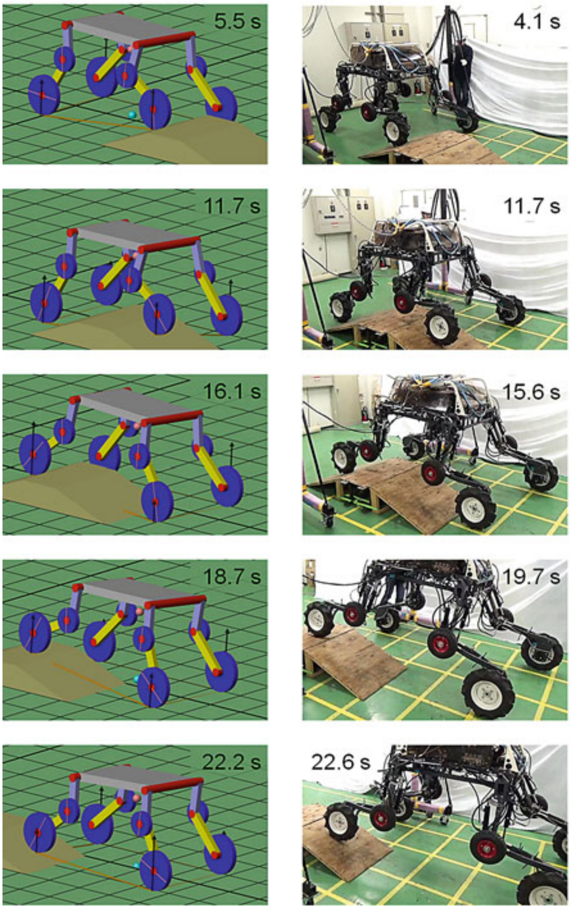
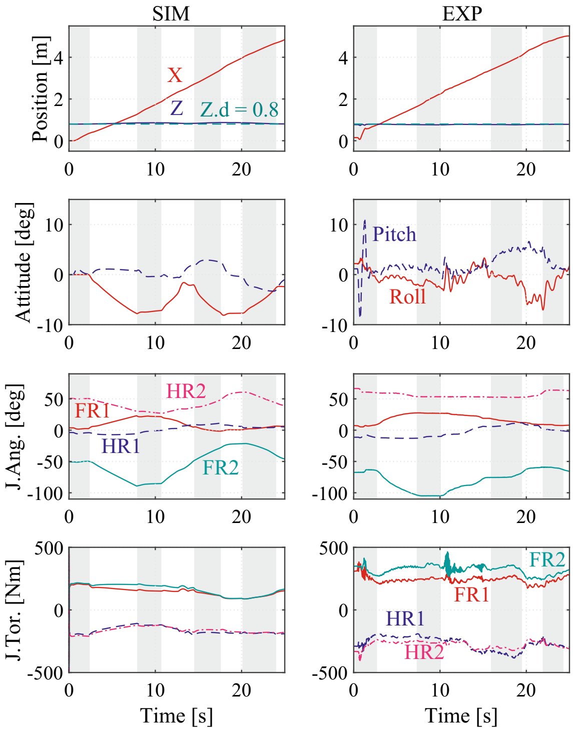


Fig. 7 Slope climbing simulation and experiment. The robot moves from left to right

To investigate the difference between the torque control and speed control for the wheel, we conducted a similar slope simulation. The simulation result with $\dot{x}_C = 0.2$ m/s is shown in Fig. 9. There are significant differences in the joint angle and torque, but the CoM and the attitude tracking performance are almost the same. Also, note that the joint angles return to a similar value.

Finally, we show the performance with the position-controlled case, where the base attitude is controlled by manipulating the height of the leg position. As can be seen from the left column of Fig. 10, the position control method could not accommodate the geometric change of the ground. Actually, the second and bottom panels show one of the driving wheels loses contact with the ground. Contrarily, in the torque-controlled method shown in the right column, no such contact loss is found.

Fig. 8 Time history of the states during the slope climbing test for simulation (left column) and experiment (right column)



5 Conclusion

This paper presented the development four-legged, four-wheel-drive hydraulic-powered rovers. By controlling the joints' torque directly using hydraulic actuators, the contact forces at their rolling contact points can be optimally distributed according to the rover's states. This allowed the acceleration of the rover's center of gravity and posture to be controlled while accommodating the terrain. This paper described the mechanical design and control system of the rover, as well as the preliminary experiments. Although the torque control for the driving wheels is not yet successfully implemented, the experiments using two rover prototypes imply the promising technological capacity of torque-controlled rovers with low-cost hydraulic components.

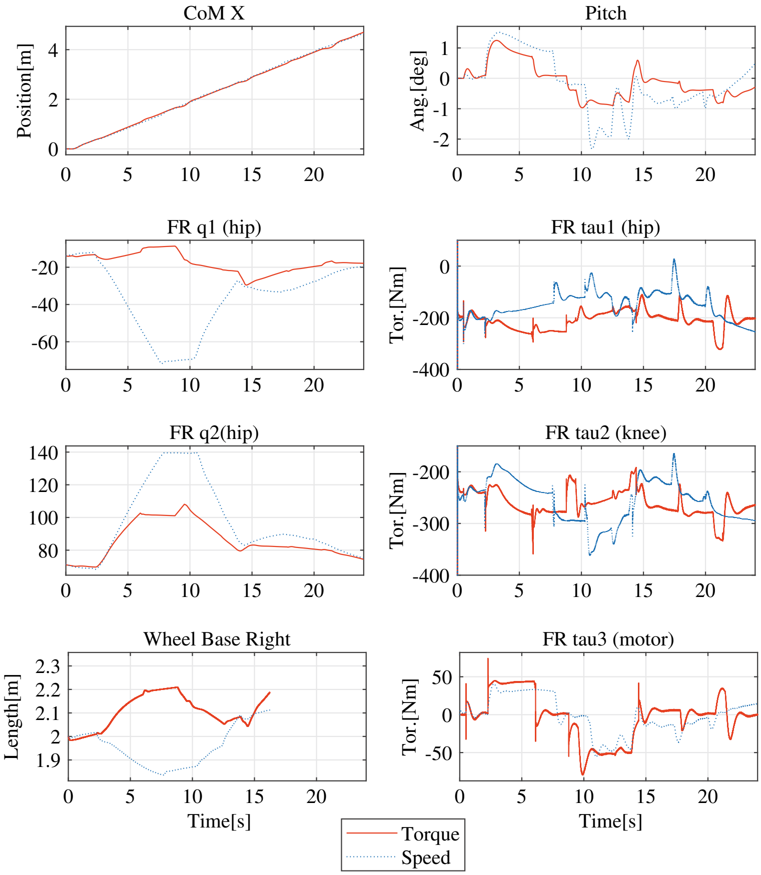


Fig. 9 Simulation comparison of torque control versus speed control for the driving wheels

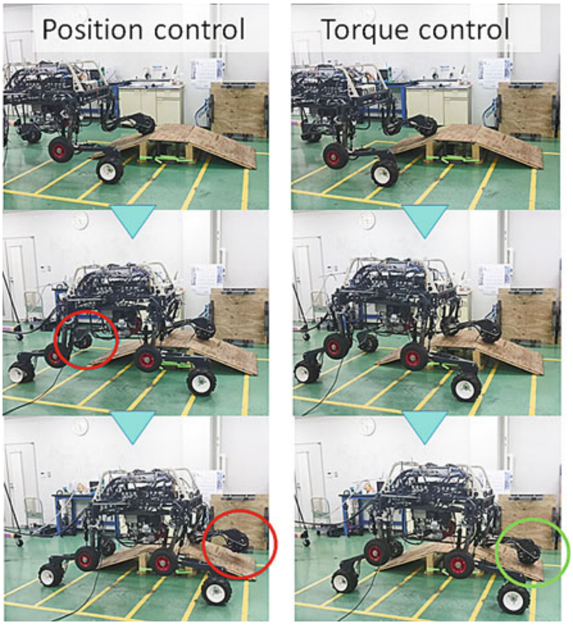


Fig. 10 Slope experiment on HYROVER-II: position control versus force control

Acknowledgements The author thanks K. Sakakibara and Y. Katayama for their support on the experiments.

References

1. NASA, Mars Exploration Rovers. https://mars.nasa.gov/mer/
2. Hirose, S.: Variable constraint mechanism and its application for design of mobile robots. Int. J. Rob. Res. **19**(11), 1126–1138 (2000)
3. Muir, P.F., Neumann, C.P.: Kinematic modeling of wheeled mobile robots. J. Robot. Syst. **4**(2), 282–340 (1987)
4. Iagnemma, K., Rzepniewski, A., Dubowsky, S.: Control of a robotic vehicles with actively articulated suspensions in rough terrain. Auton. Robots **14**, 5–16 (2003)
5. Tarokh, M., McDermott, G.J.: Kinematics modeling and analyses of articulated rovers. IEEE Trans. Robot. **21**(4), 539–553 (2005)
6. Grand, C., Benamar, F., Plumet, F.: Motion kinematics analysis of wheeled-legged rover over 3D surface with posture adaptation. Mech. Mach. Theory **45**, 477–495 (2010)
7. Taroukh, M., Ho, H., Bouloubasis, A.: Systematic kinematics analysis and balance control of high mobility rovers over rough terrain. Robot. Auton. Syst. **61**, 13–24 (2013)
8. Wilcox, B.H., Litwin, T., Biesiadecki, J., Matthews, J., Heverly, M., Morrison, J., Townsend, J., Ahmad, N., Sirota, A., Cooper, B.: Athlete: a cargo handling and manipulation robot for the moon. J. Field Robot **24**(5), 421–434 (2007)
9. Cordes, F., Babu, A., Kirchner, F.: Static force distribution and orientation control for a rover with an actively articulated suspension system. In: IEEE International Conference on Intelligent Robots and Systems, pp. 5219–5224 (2017)

10. Reid, W., Perez-Grau, F.J., Goktogan, A.H., Sukkarieh, S.: Actively articulated suspension for a wheel-on-leg rover operating on a Martian analog surface. In: IEEE International Conference on Robotics and Automation, pp. 5596–5602 (2016)
11. Hutter, M., Leemann, P., Hottiger, G., Figi, R., Tagmann, S., Rey, G., Small, G.: Force control for active chassis balancing. IEEE/ASME Trans. Mechatron. **22**(2), 613–622 (2017)
12. Ishigami, G., Nagatani, K., Yoshida, K.: Slope traversal controls for planetary exploration rover on sandy terrain. J. F. Robot. **26**(3), 264–286 (2009)
13. Hyon, S., Ida, Y., Ishikawa, J., Hiraoka, M.: Whole-body locomotion and posture control on a torque-controlled hydraulic rover. IEEE Robot. Autom. Lett. https://doi.org/10.1109/LRA.2019.2926661 (2019)
14. Hyon, S., Hale, J.G., Cheng, G.: Full-body compliant human-humanoid interaction: balancing in the presence of unknown external forces. IEEE Trans. Robot. **23**(5), 884–898 (2007)
15. Hyon, S. (2009) Compliant terrain adaptation for biped humanoids without measuring ground surface and contact forces. IEEE Trans. Robot. **25**(1), 171–178 (2009)

VI-SLAM for Subterranean Environments

Andrew Kramer, Mike Kasper, and Christoffer Heckman

1 Introduction

Mobile field robots are being deployed in ever more challenging environments. They are expected to operate in conditions where the sensors normally used for state estimation are severely limited at best and completely useless in extreme cases. As a motivating example, in the subterranean environment [1] vision is limited to what can be seen with onboard lighting and GPS is not a viable option to constrain drift. Nevertheless, autonomous subterranean robots must be able to localize and map in these challenging conditions. Without these critical skills, an autonomous robot in the subterranean environment will be unable to plan or accurately control its movements.

Vision-only SLAM is exceedingly popular [8] but this approach works well only when sensing conditions are favorable, i.e. if the environment is well-lit, and trackable features are plentiful and evenly distributed. If visual tracking fails, however, there may be no way to recover. This is especially true if visual tracking fails in an area that isn't previously explored, preventing the use of place recognition as a means to reinitialize. Generally, these methods show a great deal of promise, but are improved through the addition of other sensors in order to operate through the loss of visual tracking.

An attractive sensor combination for SLAM that balances weight, size, and cost consists of visible light cameras and inertial measurement units (IMUs). The IMU in these frameworks provides a way to localize when visual tracking fails for short periods. However, current methods for visual-inertial SLAM depend heavily on sev-

---

A. Kramer (✉) · M. Kasper · C. Heckman
University of Colorado, Boulder, USA
e-mail: andrew.kramer@colorado.edu

M. Kasper
e-mail: mike.kasper@colorado.edu

C. Heckman
e-mail: christoffer.heckman@colorado.edu

© Springer Nature Singapore Pte Ltd. 2021
G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_12

eral assumptions that do not hold in the subterranean context. The first and the most damning of these is that the environment is well and evenly lit. The second is that a large number of trackable features are evenly distributed throughout the environment. Finally, despite the community's goal of handling aggressive motions and their effect on data from these sensors, current visual-inertial SLAM methods perform best when the sensors' motion is smooth and slow, minimizing motion blur and enabling at least several informative features to be tracked [3]. For subterranean robots, it is likely that none of these conditions will be met. The only light source in the environment may be one carried by the robot. This direct lighting results in poor illumination of the environment and requires slower shutter speeds which in turn cause more motion blur. Also, depending on the environment's structure and the robot's path, the features used for tracking may not be evenly distributed in the environment. Lastly, it cannot be guaranteed that the movement of a subterranean robot will be smooth and slow. For example, a wheeled robot moving slowly over uneven, rocky ground will still experience sudden, jerky sensor motions as its wheels traverse discontinuities in the terrain.

SLAM using direct-depth sensors is also popular. 3D scanning LIDAR is a particularly attractive sensor as it does not require ambient light. It does come with its own limitations in the subterranean environment, however. For example, most LIDAR SLAM methods rely on scan matching such as iterative closest point (ICP) [2]. These methods can fail to converge to the correct relative transform between reference and target point clouds if the robot's environment is geometrically ambiguous. For instance, in a long, uniform tunnel, ICP will not have a unique solution for translation along the tunnel.

In this work, we present an evaluation of a factor graph-based sparse, indirect visual-inertial SLAM system on a novel subterranean dataset. The same dataset is run using a variety of visual measurement techniques and parameters. The effects of these variations on the SLAM system's performance are discussed with an eye to generally improving the performance of visual-inertial SLAM in the subterranean environment. We develop a novel depth-enabled framework for evaluating the triangulated position of landmarks in the visual SLAM front end and demonstrate its effectiveness in the subterranean setting. We also show the choice of parameters can have a profound effect on the system's overall performance in terms of accumulated translational drift over the robot's trajectory. Finally, we note some counterintuitive and interesting results in optimizing our system for the subterranean environment.

2 Related Work

2.1 Visual-Inertial SLAM Techniques

Visual-inertial SLAM techniques fall into two categories: filtering and batch optimization. In the filtering approach, IMU measurements are used to propagate the robot's state while image measurements are used to correct the state. The filtering approach is exemplified by MSCKF [7]. Filtering approaches to visual-inertial SLAM are generally faster and more efficient, but their accuracy is dependent on proper filter tuning. Additionally, filtering approaches only consider the previously estimated state and current sensor measurements in estimating the current state. They do not allow for the correction of previous states given new information.

In the batch optimization approach, the system's current and previous states are estimated using a camera and IMU measurements as constraints in a nonlinear optimization problem. This approach is not only more resource intensive but also more accurate than filtering as it uses measurements at multiple timesteps to jointly estimate the system's current and previous states. Of course, a batch optimization method that optimizes over the complete set of measurements from the beginning of the robot's run would quickly become computationally intractable. So there are several methods for limiting the size of the problem while maintaining an accurate state estimate. For example, keyframing assumes that most camera images do not carry significant additional information; the camera does not move significantly between frames. So keyframing approaches select a subset of the most informative camera measurements, referred to as keyframes, and only estimate the system's states from those. However, when using this technique the size of the problem still grows linearly in the number of keyframes. A sliding window filter [13] is a popular way to limit this growth. The sliding window filter optimizes over a fixed-size set of the most recent keyframes, marginalizing out measurements and states from older keyframes. This means the complexity of the SLAM problem stays roughly constant over time, allowing it to be used for long-term localization and mapping. Both of these techniques are used by the popular optimization-based visual-inertial SLAM systems OKVIS [6] and VINS-Fusion [11].

2.2 *Landmark Depth Estimation in Sparse Visual SLAM*

Several methods have been used to incorporate depth measurements from stereo or RGB-D cameras into the sparse visual SLAM problem. The most straightforward way to do this is to add the camera measurements to the optimization as a 3D constraint on the position of the keypoint in space. However, this can cause problems because the depth and projection measurements generally have different units and are subject to different kinds of errors [12]. There are a few ways to address this.

In [12], camera and depth measurements are treated separately: as a 2D measurement of a point in space projected onto the camera's sensor and a 1D measurement of that point's depth in the camera's frame. This allows for errors on these measurements to be modeled differently. It also makes depth measurements optional. A camera measurement can still be added to the optimization if its associated depth measurement can't be calculated or its uncertainty is too high.

ORB-SLAM2 takes an interesting approach to incorporate depth measurements. It extracts ORB features from the left and right stereo images. It then parameterizes camera measurements as $[u_L, v_L, u_R]$ where u_L and v_L are the horizontal and vertical coordinates of the ORB keypoint on the left camera's sensor and u_R is the horizontal coordinate of the corresponding ORB keypoint on the right camera's sensor. The depth of the keypoint follows the following linear relationship:

$$d = \frac{f_x b}{u_L - u_R} \tag{1}$$

where f_x is the camera's horizontal focal length and b is the distance between the two cameras. Because of the point's depth is a linear function of the difference in the two horizontal coordinates of the point on the camera's sensor, the horizontal coordinate of the keypoint in the right camera can be used in place of the point's depth in the optimization. This means that depth measurements in this method have the same units as projection measurements and are subject to the same kinds of errors. This method assumes, however, that the input images are stereo rectified and have the same focal length. In Sect. 3.1, we introduce a method that overcomes some of these limitations and explain how it is augmented into our overall framework.

3 VI-SLAM Method

Our SLAM approach is split into a front end for visual detection and matching and a back end for nonlinear optimization. The front end detects visual features and matches them to previously seen landmarks using the BRISK feature detector and descriptor. BRISK provides high robustness to motion blur, lighting changes, and changes in viewpoint. This makes it an ideal detector and descriptor for SLAM in subterranean environments.

Our approach's back end uses batch optimization over a sliding window [13] of recent camera frames and IMU measurements to estimate the system's state at the time of each camera frame. A camera frame consists of the 2D measurements of landmarks projected into the cameras' sensors and optionally the 1D depth of the same landmarks. An IMU measurement is the set of raw IMU readings that occur between successive camera frames. These IMU readings are preintegrated as in [4]. The sliding window consists of the most recent P camera frames linked by IMU measurements (these are referred to as IMU frames) and the N most recent keyframes. The number of IMU frames and keyframes, P and N, can be tuned to suit the robot's environment and the capabilities of its computing hardware.

The IMU frames are used to estimate robot poses, landmark positions, robot speeds, and IMU biases. After a frame passes out of the IMU window, the system decides if that frame should be a keyframe. If so, the IMU measurements are marginalized but the camera measurements continue to be used to estimate the robot's poses and the landmark positions. If not, then the camera *and* IMU measurements associated with that frame are marginalized. This marginalization strategy is pictured below in Fig. 1.

Fig. 1 Illustration of the marginalization strategy used by our SLAM method. In this case there are 2 IMU frames and 2 keyframes

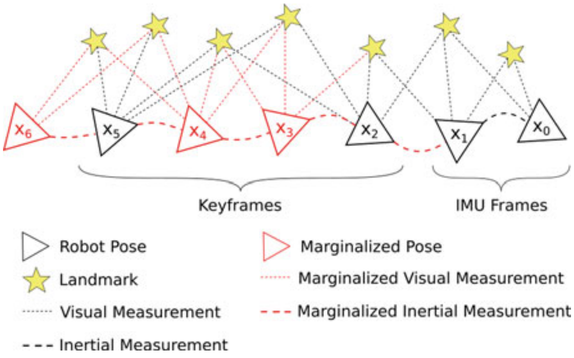
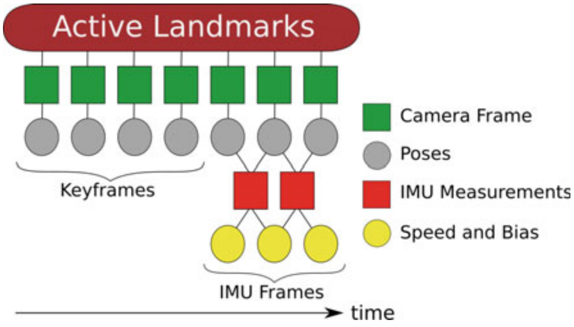

x6
x5
x4
x3
x2
x1
x0
Keyframes
IMU Frames
Robot Pose
Landmark
Visual Measurement
Inertial Measurement
Marginalized Pose
Marginalized Visual Measurement
Marginalized Inertial Measurement

Fig. 2 Diagram of the factor graph used by our VI-SLAM system


Active Landmarks
Camera Frame
Poses
IMU Measurements
Speed and Bias
Keyframes
IMU Frames
time

The graph structure of the problem is shown in Fig. 2. This problem structure is similar to that used by OKVIS [6].

The cost function used by our method's back end is defined in Eq. 2

$$J(x) := \underbrace{\sum_{i=1}^{N+P} \sum_{j \in \mathcal{J}(i)} \left[e_r^{i,j^T} W_r^{i,j} e_r^{i,j} + \left(w_d^{i,j} e_d^{i,j} \right)^2 \right]}_{\text{visual term}} + \underbrace{\sum_{i=1}^{P} e_s^{i^T} W_s^i e_s^i}_{\text{inertial term}} \tag{2}$$

where N is the number of keyframes and P is the number of IMU frames in the optimization, $\mathcal{J}(i)$ is the set of landmarks observed in frame i, $e_r^{i,j}$ is the reprojection error for landmark j in frame i, $e_d^{i,j}$ is the scalar depth error for landmark j in frame i, e_s^i is the IMU error for frame i, and the W matrices contain the weights for the errors. Note that inertial terms are only present for the IMU frames, and depth terms (e_d and w_d) are optional for all frames.

Our SLAM system uses three steps to ensure robustness to incorrect data association. First, the system's front end uses IMU measurements to propagate the camera's pose from the last optimized pose to the current frame time and predicts where matched landmarks should be observed in the new pose. Landmark matches that are greater than a certain distance from this predicted observation are discarded. Second, the front end uses RANSAC to ensure the observed positions of the matched landmarks in the camera frame are geometrically consistent with the landmarks' estimated positions in the global frame. Lastly in the system's back end, the optimizer computes the cost using robust norms.

3.1 Direct Depth Measurement

In subterranean environments with poor lighting, there are often very few trackable features available. To make matters worse, the features are often poorly distributed. For example, if the robot is carrying its own light, nearby objects will be well illuminated but distant objects will often be very poorly illuminated. So the few landmarks that can be tracked by the SLAM system will be very close to the camera. This can lead to ambiguities in monocular visual tracking because nearby points are less informative for estimating the robot's orientation. The inverse is also true, when the tracked landmarks are all distant from the camera they are less informative for estimating the robot's translation. Finally, scale is only introduced through the optimization of IMU factors, which makes it critically dependent on the accuracy of the IMU.

One solution to these problems is to use multi-camera systems. If the cameras have significant overlap in their fields of view, they can provide multiple measurements of the same landmark at the same timestep but from different viewpoints. This can help to constrain the estimated distance to that landmark. However, this method treats measurements from multiple cameras the same as monocular measurements: the measurements are simply the 2D coordinates of a landmark when projected onto each camera's sensor. This strategy does not take advantage of the fact that a direct depth measurement can be obtained for any landmark that is visible in both cameras.

Our method does take advantage of this fact. It first uses brute-force matching to match keypoints between the left and right stereo images. It then uses stereo triangulation to calculate the distance to the point in space corresponding to the keypoint. Two measurements are then added to the optimization problem: a 2D camera measurement of the point in camera zero's frame; and a 1D measurement of the point's distance from the left camera's sensor as in [12].

We keep the depth and 2D camera measurements separate for two reasons. First, it allows us to weight errors and handle outliers for each measurement type differently. Second, treating projection and depth as separate measurements means that depth measurements can be used but they are not required. So in our system, keypoints for which depth cannot be calculated are still added to the system as 2D measurements

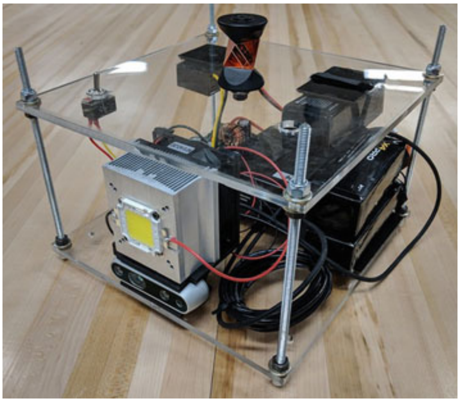


Fig. 3 The sensor rig used to capture our subterranean dataset. Includes an Intel RealSense D435 stereo camera, a Lord Microstrain 3DM-GX5-15 IMU, and a 9000 lumen LED headlamp

without depth. This allows our system to use depth measurements when they're available but the tracking does not fail when depth measurements aren't available. Instead, it degrades gracefully back to using only 2D measurements.

4 Experiments

4.1 Sensor Setup

To obtain our dataset, we used the infrared stereo cameras on an Intel RealSense D435 and a Lord Microstrain 3DM-GX5-15 IMU as sensors. Images were captured at 640 × 480 resolution and IMU messages were read at 100 Hz. The performance of a visual-inertial SLAM system is highly dependent on knowing the camera-to-IMU transform (referred to as the extrinsics) with great accuracy. To obtain the extrinsics as well as the intrinsics (focal length and distortion parameters) for both cameras, we used Vicalib, a calibration library based on ceres solver. Vicalib first finds intrinsic camera parameters (focal length, central point, and distortion parameters) by tracking conics on a calibration target and solving an overdetermined perspective-n-point ransac problem through gradient descent. Vicalib then determines extrinsic parameters (camera-to-IMU rotation and translation) between each IR camera and the IMU using gradient descent with an IMU residual as presented in [9]. In this optimization, intrinsic parameters are fixed and only the extrinsic parameters are varied. Because the data was taken in complete darkness in a subterranean environment, the sensor rig was also equipped with a forward-facing 9000 lumen soft-white LED headlamp (Fig. 3).

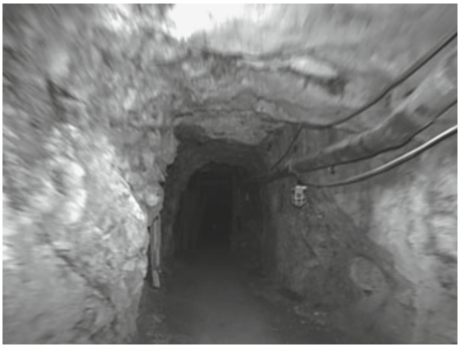


Fig. 4 Typical camera image from the mine dataset with the headlamp at 15% intensity. Note the scene is well lit near the camera, and almost completely dark just a few meters from the camera. This is typical for the entire dataset

4.2 Dataset Description

Our dataset was recorded in the Hidee Gold Mine in Central City, Colorado. It consists of camera images and IMU messages recorded by our handheld sensor rig as it was carried from the start to the end of the mine and back. The total distance covered was roughly 340 m at an average speed of 1.4 m/s. The environment consisted of tunnels roughly 2 m in height and 1.5 m wide with bare rock walls. An example image of the environment can be seen in Fig. 4.

There was no ambient light in the tunnels. The only available light in the environment came from the sensor rig's onboard LED headlamp. Two different runs were done over the same course, one with the headlamp set to 50% intensity and one with the headlamp at 15% intensity. Apart from the headlamp setting, the two runs are nearly identical. The limited, direct lighting from the headlamp had two major effects. First, the cameras needed to use long exposure times to adequately expose their images. This means the images from the two IR cameras often have severe motion blur as pictured in Fig. 5.

Second, light intensity in the scene generally decreases with the square of the distance from the headlamp. This means parts of the scene that are very close to the camera are well exposed, but parts that are even a moderate distance from the camera are very underexposed. This gives the camera an artificially narrow depth of field. The result is that the SLAM system was unable to track keypoints more than a few meters from the cameras. This is illustrated in Fig. 6.

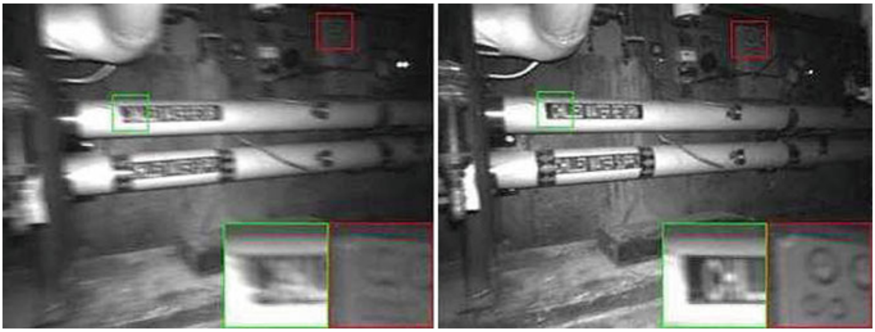


Fig. 5 Camera image comparing motion blur with headlamp at 15% intensity (left image) to 50% intensity (right image). Average image intensity for both images is similar but the left image has significantly more motion blur

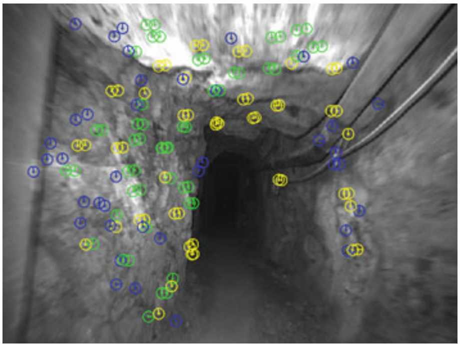


Fig. 6 Example stereo image demonstrating the limited range of the sensor rig's headlamp and the consequently shallow depth range in which keypoints are reliably trackable

4.3 Test Description

Each dataset was run in our visual-inertial SLAM system with a different set of measurement parameters, referred to as a setting. Specifically, the keypoint detection threshold, keypoint matching threshold, and stereo measurement techniques were varied. The detection threshold is the minimum strength a keypoint must have before it is added as a measurement to the SLAM front end [5]. The matching threshold is the maximum hamming distance between two keypoint descriptors that can be considered a match. Lastly, stereo measurements can be added to the SLAM system in one of two ways. The conventional method is to add measurements from multiple cameras as multiple, independent constraints on 2D reprojection error as is done in OKVIS [6]. The alternative is to use the method described in Sect. 3.1, which matches features between the two cameras and explicitly calculates depth measurements through stereo triangulation. The parameters used in each setting we tested are given in Table 1.

Table 1 Parameters used for each setting on which the SLAM system was tested

Setting number	1	2	3	4	5	6	7	8	9	10	11	12
Detection threshold	50	50	50	50	40	40	40	40	40	40	40	40
Matching threshold	80	80	80	80	60	60	60	60	80	80	80	80
Headlamp setting	Low	High	High	Low	Low	High	Low	High	Low	High	Low	High
Depth measurement	No	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes

4.4 *Error Evaluation*

Due to the length of the path, the confined space of the mine tunnels, and the lack of line-of-sight between all points in the path, we were not able to obtain accurate groundtruth over the whole path with motion capture, laser tracker, or similar system. Instead, SLAM performance is evaluated only on the accumulated translational drift over the course of the dataset. To assist in calculating this metric, we started and ended each dataset in the same place and placed an AprilTag [10, 14] at that location. This allows us to calculate the groundtruth transform from the start of the dataset to the end, $T_{\text{groundtruth}}$. We then compare this to the same quantity estimated by the SLAM system, T_{est} as follows:

$$T_{\text{err}} = T_{\text{est}} T_{\text{groundtruth}}^{-1} \tag{3}$$

The error e is defined as the magnitude of the translational component of T_{err}: $e = \|t_{\text{err}}\|_2$.

5 Results

Table 2 shows the averaged results over 5 runs for each setting. Each run was 340 m at an average speed of 1.4 m/s. 2D keypoint counts are not reported for settings in which depth measurements are used because the keypoint detection and matching parameters are the same as in other settings where depth measurements were not used. So the 2D keypoint counts for settings with depth would be redundant with the corresponding settings without depth measurements. Also note the 2D and depth point counts refer to the number mean number of points detected per camera frame .

Figure 7 shows the whole range of translational errors obtained for each setting. Settings that use the brighter headlamp are plotted in blue while low-light settings are plotted in red. It is interesting to note that low-light settings tend to have lower translational error and are more consistent than the high-light settings. Figure 8 shows

Table 2 Averaged results for all settings

Setting number	1	2	3	4	5	6	7	8	9	10	11	12
Translational error (m)	18.8	12.7	36.3	16.0	12.0	29.2	9.72	23.9	12.3	29.6	11.8	34.9
2D points detected	88.5	128			122	180			119	181		
2D points matched	43.3	58.5			53.6	70.8			56.6	84.0		
Depth measurements			41.2	28.7			39.5	59.2			40.0	59.1
Depth matches			20.6	14.9			18.4	24.7			21.1	28.0

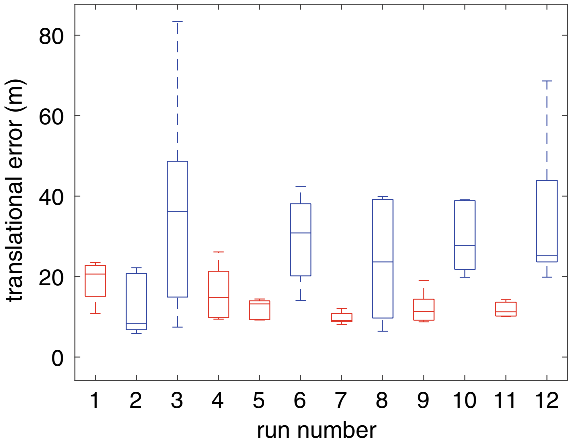


Fig. 7 Accumulated translational error for all settings. Red plots represent settings with low light. Blue plots represent settings with high light

the same data grouped into settings with high and low light and with and without depth measurement. Curiously, depth measurements are not helpful when used in high-light settings. However, in low-light settings, depth measurements do improve both the best-case translational drift and the variance in drift. Also, the low-light settings in general are better and more consistent than high-light settings.

As expected, higher keypoint detection thresholds result in fewer total detected keypoints and depth measurements per frame, and higher matching thresholds result in more measurements matched to existing landmarks per frame. Also, more keypoints and depth measurements are able to be detected and matched when the headlamp setting is high. This increase in the number of points detected and matched does not translate to better performance in terms of translational drift, however. The settings that consistently had the best performance, settings 7 and 5, were done in low light with low detection and matching thresholds.

Lastly, Fig. 9 shows plots of the estimated positions of the sensor rig throughout two example runs. Figure 9a shows the best results obtained and Fig. 9b shows the worst results obtained. Only the estimated positions in the x and y dimensions are

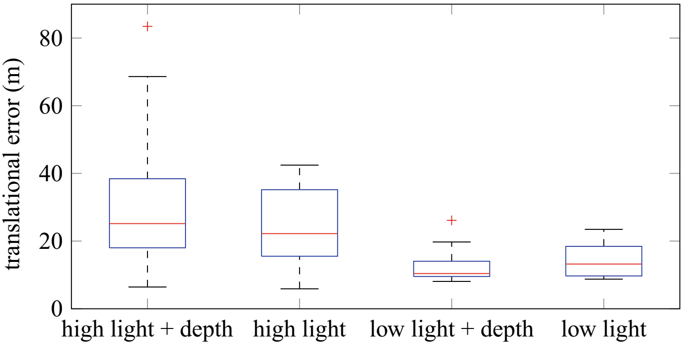
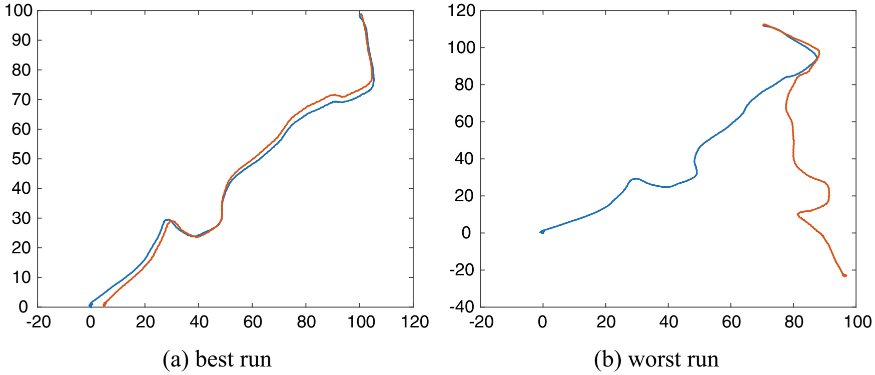


Fig. 8 Translational errors for high-light and low-light settings. Shows settings for which direct depth measurements were used separately from those for which depth wasn't used



(a) best run

(b) worst run

Fig. 9 The estimated positions for two example runs plotted in the $x - y$ plane. The outgoing paths are plotted in blue and the return paths are plotted in orange. Note the outgoing and return paths in **a** overlap almost perfectly, while there is nearly no overlap in **b**

shown because the groundtruth translation and drift in the z-direction for all runs is very small.

6 Conclusions

From these results, it is clear that it is possible to obtain consistently good performance from a visual-inertial SLAM system in a confined subterranean environment with no ambient light. The best run on the best setting (setting number 7; i.e. low detection threshold, low matching threshold, low headlamp setting using direct depth measurements) had a translational drift of 5.90 m, just 1.7% of the total path distance. On the worst-performing run for that setting, the drift was still only 3.5% of the total path distance. However, obtaining such performance requires careful tuning of the front end, and the effects of each parameter can be counterintuitive. We will now describe our findings through our exploration.

Quite counterintuitively, the use of a brighter headlamp is not always helpful. The reasons for this are not completely clear, but a subjective analysis of the image streams on the high- and low-light settings gives some clues. With brighter light, motion blur is minimized and more features are detectable. However, brighter light also results in a stronger brightness dropoff from nearer to more distant objects in the scene as a result of autoexposure. While this could be mitigated through the careful tuning of autoexposure control schemes or focusing of the light source with reflectors or lenses, these strategies would likely be highly geometry and appearance dependent. In short, brighter light settings can actually result in a narrower depth of field in which features are detectable. Dimmer light settings, meanwhile, result in a larger depth of field over which features can be detected and tracked. So even though a dimmer headlamp results in fewer features and more blurring, as long as the blur is not overwhelming all salient components of the image, those features are better distributed throughout the scene and are therefore more informative for the visual tracking front end of SLAM. Furthermore, analysis of the raw depth measurements taken over two runs with bright and dim lighting supports the idea that a dimmer light setting can result in better distributed keypoints. The mean depth of detected keypoints was 2.12 m for dim lighting and 2.01 m for bright lighting. Additionally, the depth variance of detected keypoints was 0.68 for bright lighting and 0.72 for dim lighting. So in dim light, the detected keypoints are slightly deeper and more widely distributed.

The use of direct depth measurements can be helpful but this is not always the case. In the bright -lit settings, the use of depth measurements did not affect the system's performance significantly, and the best and the most consistent results with bright light were obtained without using depth measurements. In low-light settings, however, the use of depth measurements usually resulted in better and more consistent performance. This is likely related to the brighter light's effect on feature distribution in the camera's field of view. In brighter light settings, the detected features are all close to the camera. In this case, a small amount of translation of the camera results in a large difference in the feature's projected position on the camera's sensor, and it is easy to estimate the feature's position in 3D space from 2D measurements. So the depth measurement does not add much additional information. With dimmer light, on the other hand, the features may be further from the camera and, due to higher motion blur and changes in illumination, the features may not be tracked for long enough to get a good estimate of their depth from 2D measurements alone. In this case, direct depth measurements can add significant additional information.

To conclude, this work illuminates some of the challenges inherent in using visual-inertial SLAM in the subterranean environment with no ambient light. These challenges are often the result of direct lighting from an onboard light source. The use of an onboard light source causes a high brightness gradient throughout the scene, forcing the operator to make trade-offs between the number and quality of features that can be detected. This work shows it is possible to obtain good visual-inertial

SLAM performance in the subterranean environment with careful tuning. The need for this environment-specific tuning could be lessened if the effects of direct lighting from an onboard light source were taken into account in the system's measurement model, however, and future work should be directed toward this goal.

References

1. Darpa subterranean (subt) challenge. https://www.darpa.mil/program/darpa-subterranean-challenge
2. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992.) http://dblp.uni-trier.de/db/journals/pami/pami14.html#BeslM92
3. Carlone, L., Karaman, S.: Attention and anticipation in fast visual-inertial navigation. IEEE Trans. Robot. **35**(1), 1–20 (2019)
4. Forster, C., Carlone, L., Dellaert, F., Scaramuzza, D.: On-manifold preintegration for real-time visual–inertial odometry. IEEE Trans. Robot. **33**(1), 1–21 (2017). ISSN 1552-3098, 1941-0468. https://doi.org/10.1109/TRO.2016.2597321. https://ieeexplore.ieee.org/document/7557075/
5. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: binary robust invariant scalable keypoints. In: Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pp. 2548–2555. Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. https://doi.org/10.1109/ICCV.2011.6126542
6. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.T.: Keyframe-based visual–inertial odometry using nonlinear optimization. Int. J. Robot. Res. (IJRR) (2016)
7. Mourikis, A.I., Roumeliotis, S.I.: A multi-state constraint kalman filter for vision-aided inertial navigation. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 3565–3572 (2007)
8. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE Trans. Robot. **31**, 1147–1163 (2015)
9. Nobre, F., Heckman, C.R., Sibley, G.T.: Multi-sensor slam with online self-calibration and change detection, pp. 764–774, 03 2017. ISBN 978-3-319-50114-7. https://doi.org/10.1007/978-3-319-50115-4_66
10. Olson, E.: AprilTag: a robust and flexible visual fiducial system. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3400–3407. Shanghai, China, May 2011, IEEE. ISBN 978-1-61284-386-5. https://doi.org/10.1109/ICRA.2011.5979561. http://ieeexplore.ieee.org/document/5979561/
11. Qin, T., Pan, J., Cao, S., Shen, S.: A general optimization-based framework for local odometry estimation with multiple sensors (2019)
12. Scherer, S., Dubé, D., Zell, A.: Using depth in visual simultaneous localisation and mapping. In: 2012 IEEE International Conference on Robotics and Automation, pp. 5216–5221 (2012)
13. Sibley, G.: A sliding window filter for slam (2006)
14. Wang, J., Olson, E.: AprilTag 2: efficient and robust fiducial detection. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4193–4198. Daejeon, South Korea, October 2016. IEEE. ISBN 978-1-5090-3762-9. https://doi.org/10.1109/IROS.2016.7759617. http://ieeexplore.ieee.org/document/7759617/

Simultaneous Localization and Mapping of Subterranean Voids with Gaussian Mixture Models

Wennie Tabib and Nathan Michael

1 Introduction

Search and rescue operators benefit from enhanced situational capabilities provided by robots during time-critical and life-threatening operations [19]. Robot perception in disaster environments is challenging as these environments are highly cluttered, unstructured, and unpredictable. In addition, communication infrastructure may deteriorate or become completely disabled over extensive areas for hours or days during natural disasters [20] hindering search and rescue operations. These environmental constraints create the need for compact, efficient environment representations that are transmissible over low-bandwidth communication networks.

GMMs compactly represent high-resolution sensor observations and have been demonstrated to enable occupancy modeling [21] and estimate pose [26] with orders of magnitude reduction in memory required to store and transmit the data as compared to raw sensor observations [27]. While Eckart [10] provides a qualitative evaluation of a laser-based SLAM formulation leveraging hierarchical GMMs, to the best of the authors' knowledge, a quantitative analysis of GMM-based SLAM that incorporates loop closures has not been conducted. The proposed work bridges this gap in the state of the art by developing GMM mapping that incorporates global consistency and exploits the sparse compact support to reduce the dimensionality of calculations.

The paper is organized as follows: Sect. 2 details related work followed by a description of the methodology in Sect. 3. Section 4 provides an analysis of the proposed approach as compared to the state of the art and Sects. 5 and 6 conclude the paper with a discussion of the limitations of the method and future work.

W. Tabib (✉) · N. Michael
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: wtabib@cmu.edu

N. Michael
e-mail: nmichael@cmu.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_13

2 Related Work

Generative modeling has experienced a resurgence in recent years due to the vision that disparate point cloud-based perception algorithms may be unified into a common pipeline [10]. A top-down hierarchical GMM approach is developed by Eckart et al. [8] to accelerate learning by employing a sparsification technique that adjusts the posterior between levels so that child mixture components share geometric context information in a soft partitioning scheme. The proposed approach does not employ a hierarchy but leverages the Mahalanobis distance as a sparsification technique in Expectation Maximization. Srivastava and Michael [23] develop a bottom-up hierarchical approach that assumes known pose estimates and merges partial sensor observations into one monolithic GMM. This approach is susceptible to accumulation of noise and pose drift, which makes corrections to enable global consistency difficult. In contrast, the proposed approach represents each sensor observation as a GMM to enable corrections when closing the loop.

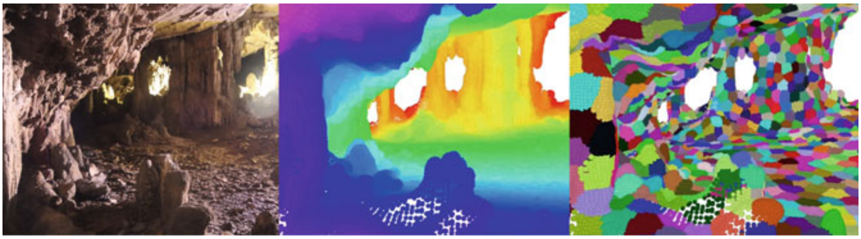
The Normal Distribution Transform map is learned by voxelizing a sensor observation and calculating a Gaussian density within each cell [24]. Pose is estimated by minimizing the L2 distance between the two distributions. While being fast, this approach has been determined to be less accurate than point-based approaches like Generalized-Iterative Closest Point (GICP) [26].

Evangelidis and Horaud [12] develop a batch registration algorithm for multiple point sets that estimates GMM parameters, rotations, and translations via an Expectation Maximization algorithm. While being accurate, it is not real-time viable. Eckart et al. [9] develop an approach that simultaneously trains and registers GMMs. The Mahalanobis distance is used to compute the distance between points and estimate optimal rotation and translation for registration. While the accuracy is on par with several ICP variants, the approach is evaluated on an NVIDIA Titan X GPU, which is prohibitive for use on size, weight, and power-constrained systems.

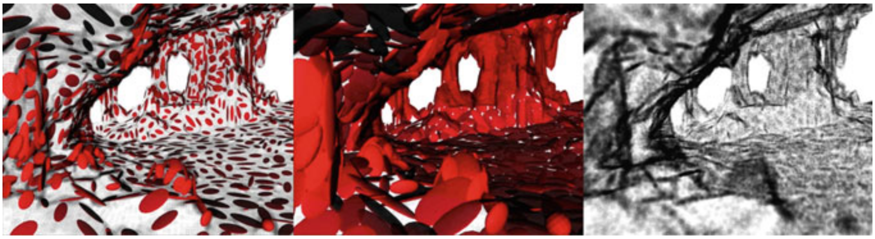
Behley and Stachniss [2] develop a SLAM formulation for 3D laser range data, but require a GPU to achieve real-time results. The authors opt for a frame-to-model ICP that registers a point cloud to a surfel model. Loop closure detection proceeds by checking nearby poses against the current laser scan with ICP and renders a view of the current map to determine if the loop closure leads to a consistent map given the current scan. Multiple initializations of ICP with different translations and rotations are required to determine an adequate pose estimate. In contrast to ICP, GICP has been demonstrated to be more robust to large changes in rotation and translation [22]. The proposed approach is compared to [2].

3 Methodology

In this work, each sensor observation is represented as a GMM. Figure 1 illustrates the method by which a GMM is learned from a sensor observation to generate an environment map. A mathematical description of the GMM and the approach



(a) Image of Rapps Cave (b) Pointcloud representation (c) K-Means++ clusters



(d) 1-sigma covariances (e) 2-sigma covariances (f) Resampled model

Fig. 1 Overview of the process to learn a GMM from a sensor observation. **a** An image taken from Rapps Cave in WV, USA. **b** A corresponding point cloud colored according to viewing distance (red is further away). **c** Each color corresponds to a cluster learned with the K-Means++ algorithm. The red ellipsoids in **d** and **e** are visualizations of the 1-sigma and 2-sigma contours of the constant probability of the Gaussian mixture components after running EM, respectively. The 1-sigma visualization represents approximately 20% of the probabilistic coverage of the underlying point density and 2-sigma approximately 74%. Because the GMM is a generative model, samples may be drawn from the distribution to obtain the reconstruction in **f**

to enable real-time viable parameter estimation are detailed in Sect. 3.1. Section 3.2 describes the distribution-to-distribution registration, SLAM, and loop closure approaches.

3.1 Gaussian Mixture Model

A GMM is a probability distribution that represents multivariate data as a weighted combination of M Gaussian distributions. The probability density of the GMM is represented as

$$p(\boldsymbol{x} \mid \boldsymbol{\xi}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$$

where $\boldsymbol{x} \in \mathbb{R}^D$, π_m is a weight such that $0 \leq \pi_m \leq 1$, $\sum_m^M \pi_m = 1$, and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda})$ is a D-dimensional Gaussian density function with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Lambda}$.

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \frac{|\boldsymbol{\Lambda}|^{-1/2}}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

The parameters of the distribution are compactly represented as $\boldsymbol{\xi} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m\}_{m=1}^M$. Estimating the parameters of a GMM remains an open area of research [14]. Given the density function $p(\boldsymbol{x}|\boldsymbol{\xi})$ and observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\boldsymbol{x} \in \mathbb{R}^D$ assumed to be independent and identically distributed with distribution p, the density for the samples is

$$p(\boldsymbol{X}|\boldsymbol{\xi}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\xi}) = \mathcal{L}(\boldsymbol{\xi}|\boldsymbol{X})$$

where $\mathcal{L}(\boldsymbol{\xi}|\boldsymbol{X})$ is called the likelihood function and the goal is to find the $\boldsymbol{\xi}^*$ that maximizes $\mathcal{L}$ [3]

$$\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} \ \mathcal{L}(\boldsymbol{\xi}|\boldsymbol{X})$$

It is analytically easier to maximize $\ln(\mathcal{L}(\boldsymbol{\xi}|\boldsymbol{X}))$, but the presence of the summation over m inside the logarithm makes $\boldsymbol{\xi}$ difficult to compute, and taking the derivative of this log likelihood function and setting to zero is made intractable because the resulting equations are no longer in closed form [4]. Instead, latent variables $b_{nm} \in \boldsymbol{B}$ are introduced that take a value of 1 if the sample $\boldsymbol{x}_n$ is in cluster m and 0, otherwise (called a 1-of-M coding scheme). A new likelihood function is defined, $p(\boldsymbol{X}, \boldsymbol{B}|\boldsymbol{\xi}) = \mathcal{L}(\boldsymbol{\xi}|\boldsymbol{X}, \boldsymbol{B})$, called the complete data likelihood.

The Expectation step in Expectation Maximization finds the expected value of $\ln p(\boldsymbol{X}, \boldsymbol{B}|\boldsymbol{\xi})$ by the following function [3]:

$$Q(\boldsymbol{\xi}, \boldsymbol{\xi}^i) = E\Big[\ln p(\boldsymbol{X}, \boldsymbol{B}|\boldsymbol{\xi})|\boldsymbol{X}, \boldsymbol{\xi}^i\Big]$$

The Maximization step maximizes the expectation of the previous equation:

$$\boldsymbol{\xi}^{i+1} = \arg\max_{\boldsymbol{\xi}} Q(\boldsymbol{\xi}, \boldsymbol{\xi}^i)$$

Each iteration of these steps is guaranteed to increase the log likelihood and ensure that the algorithm converges to a local maximum of the likelihood function [3].

3.1.1 Initialization

Kolouri et al. [18] find the EM algorithm to be sensitive to the choice of initial parameters and Jian and Vemuri [15] prove that random initialization causes the EM algorithm to converge to a bad critical point with high probability. Because the EM algorithm does not guarantee convergence to a global optimum, the initialization is critical for convergence to a good stationary point. The proposed approach implements the K-Means++ algorithm [1], which is an unsupervised learning algorithm that provides an initial clustering of the sensor data (see Fig. 1c) and has advantages over the standard K-Means algorithm in that it provides approximation guarantees for the optimality of the algorithm that improve the speed and accuracy. Several variants of K-Means are proposed in the literature [6, 11, 13], but this work leverages the method proposed in Elkan [11] as it was found to achieve the best performance. Elkan [11] increases efficiency by employing the triangle inequality and maintaining lower and upper bounds on distances between points and centers. Given the range sensor data in Fig. 1b, the K-Means++ algorithm outputs the clustering shown in Fig. 1c. These clusters are used to seed the EM algorithm detailed in the next section.

3.1.2 Expectation Maximization

The EM algorithm proceeds with the following steps:

1. Initialize $\boldsymbol{\mu}_m$, $\boldsymbol{\Lambda}_m$, and π_m with the method detailed in Sect. 3.1.1.
2. **E step**. Evaluate the responsibilities γ_{nm} using the current parameters $\boldsymbol{\mu}_m$, $\boldsymbol{\Lambda}_m$, and π_m:

$$\gamma_{nm} = \frac{\pi_m \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)}{\sum_{j=1}^{M} \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j)} \tag{1}$$

3. **M step**. Estimate the new parameters $\boldsymbol{\mu}_m^{i+1}$, $\boldsymbol{\Lambda}_m^{i+1}$, and π_m^{i+1} using the current responsibilities, γ_{nm}.

$$\boldsymbol{\mu}_m^{i+1} = \frac{\sum_{n=1}^{N} \gamma_{nm} \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_{nm}} \tag{2}$$

$$\boldsymbol{\Lambda}_m^{i+1} = \frac{\sum_{n=1}^{N} \gamma_{nm} (\boldsymbol{x}_n - \boldsymbol{\mu}_m^i)(\boldsymbol{x}_n - \boldsymbol{\mu}_m^i)^T}{\sum_{n=1}^{N} \gamma_{nm}} \tag{3}$$

$$\boldsymbol{\pi}_m^{i+1} = \sum_{n=1}^{N} \frac{\gamma_{nm}}{N} \tag{4}$$

4. Evaluate the log likelihood

$$\ln p(\boldsymbol{X}|\boldsymbol{\xi}) = \sum_{n=1}^{N} \ln \left(\sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m) \right) \tag{5}$$

   and check for convergence of either the parameters or the log likelihood. If convergence is not achieved, iterate again from step 2.

Figure 1d and e provide a visualization of the GMM with 1- and 2-sigma covariances after running EM. While the K-means algorithm performs a hard assignment of data points to clusters, the EM algorithm makes a soft assignment based on posterior probabilities. The intuition behind the soft assignment yields one of the contributions of this paper which is that because Gaussians fall off quickly, points far away from an initialized density will have a small effect on the updated parameters for that density. The responsibility matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times M}$ scales with the number of samples N and the number of components M, so to reduce the computation time, an approximation is made to ignore points that lie outside a Mahalanobis distance greater than λ for the initialized density. EM is modified in the following way:

1. Initialize $\boldsymbol{\mu}_m^1$, $\boldsymbol{\Lambda}_m^1$, and π_m^1 with the method detailed in Sect. 3.1.1.
2. For a given component m, evaluate only the γ_{nm} that satisfy Mahalanobis-bound:

$$\lambda > \sqrt{(\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)^T (\boldsymbol{\Lambda}_m^1)^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_m^1)} \tag{6}$$

3. Estimate the updated parameters $\boldsymbol{\mu}_m^{i+1}$, $\boldsymbol{\Lambda}_m^{i+1}$, and π_m^{i+1} with the current responsibilities γ_{nm} and Eqs. (2)–(4).
4. Evaluate the log likelihood (Eq. (5)) and iterate again from step 2 if convergence is not achieved.

3.2 Pose Graph SLAM via GMM Registration

Each sensor observation is represented as a GMM with M components and successive GMMs are registered together using the approach detailed in [26], which is summarized in Sect. 3.2.1. Loop closures are detected by aligning observations within a given radius r of the current pose until a match is found that achieves the fitness threshold α.

The pose graph is formulated as a factor graph [16] where factors represent constraints between poses, or nodes. The factor graph $G = (\mathscr{F}, \mathscr{V}, \mathscr{E})$ is composed of factor nodes $f_i \in \mathscr{F}$ and variable nodes $v_j \in \mathscr{V}$ with edges $e_{ij} \in \mathscr{E}$ connecting the factor nodes and variable nodes. The factor graph finds the variable assignment $\mathscr{V}^*$ that maximizes

$$\mathscr{V}^* = \arg\max_{V} \prod_{i} f_i(V_i) \tag{7}$$

where V_i is the set of variables adjacent to the factor f_i and independence relationships are encoded by the edges e_{ij} such that each factor f_i is a function of the variables in V_i. The pose graph uses relative constraints from GMM registration with a covariance based on the depth sensor model.

3.2.1 Registration

Following the approach of [26], let $\mathscr{G}_i(\boldsymbol{x})$ and $\mathscr{G}_j(\boldsymbol{x})$ denote GMMs learned from sensor observations $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$, respectively,

$$\mathscr{G}_i(\boldsymbol{x}) = \sum_{m}^{M} \pi_m \mathscr{N}(\boldsymbol{x} | \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$$

$$\mathscr{G}_j(\boldsymbol{x}) = \sum_{k}^{K} \tau_k \mathscr{N}(\boldsymbol{x} | \boldsymbol{\nu}_k, \boldsymbol{\Omega}_k)$$

and let $T(\cdot, \theta)$ denote the rigid transformation consisting of a rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$. To register $\mathscr{G}_j(\boldsymbol{x})$ into the frame of $\mathscr{G}_i(\boldsymbol{x})$, optimal rotation and translation parameters must be found such that the squared L2 norm between the distributions $\mathscr{G}_i(\boldsymbol{x})$ and $T(\mathscr{G}_j(\boldsymbol{x}), \theta)$ is minimized. The transformation parameters θ consisting of $\boldsymbol{R}$ and $\boldsymbol{t}$ may be applied to a GMM $\mathscr{G}_j(\boldsymbol{x})$ in the following way:

$$T(\mathscr{G}_j(\boldsymbol{x}), \theta) = \sum_{k=1}^{K} \tau_k \mathscr{N}(\boldsymbol{x} | \boldsymbol{R}\boldsymbol{\nu} + \boldsymbol{t}, \boldsymbol{R}\boldsymbol{\Omega}_k \boldsymbol{R}^T)$$

The cost function is

$$\theta^* = \arg\min_{\theta} \int ||\mathscr{G}_i(\boldsymbol{x}) - T(\mathscr{G}_j(\boldsymbol{x}), \theta)||_2^2 d\boldsymbol{x} \tag{8}$$

$$= \arg\min_{\theta} \int ||\mathscr{G}_i(\boldsymbol{x})||_2^2 + ||T(\mathscr{G}_j(\boldsymbol{x}), \theta)||_2^2 - 2\mathscr{G}_i(\boldsymbol{x}) T(\mathscr{G}_j(\boldsymbol{x}), \theta) d\boldsymbol{x} \tag{9}$$

The first term in Eq. (9) does not depend on the transformation parameters θ and remains constant; therefore, it may be eliminated. The second term is invariant under rigid transformation and also may be eliminated. Thus, Eq. (8) may be rewritten as

$$\theta^* = \arg\min_{\theta} - \int 2\mathscr{G}_i(\boldsymbol{x}) T(\mathscr{G}_j(\boldsymbol{x}), \theta) d\boldsymbol{x}$$

which has a closed-form solution as shown in [26]

$$\theta^* = \arg\min_{\theta} - \sum_{m=1}^{M} \sum_{k=1}^{K} \pi_m \tau_k \mathscr{N}(\boldsymbol{\mu}_m | \boldsymbol{R}\boldsymbol{\nu}_k + \boldsymbol{t}, \boldsymbol{\Lambda}_m + \boldsymbol{R}\boldsymbol{\Omega}_k \boldsymbol{R}^T)$$

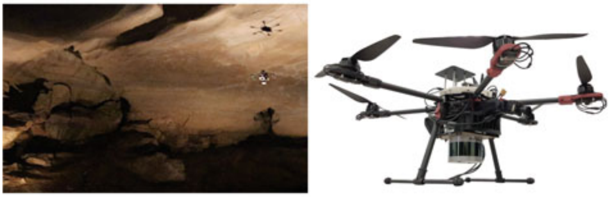
The optimal rigid transformation parameters may be solved by employing a Riemannian trust-region method with conjugate gradients [5]. The registration method used in this work is the Isoplanar variant from [26] that modifies the covariances prior to registration to smooth the cost function. The modified covariance is computed by calculating the eigen decomposition $\boldsymbol{\Lambda}_m = \boldsymbol{U}_m \boldsymbol{D}_m \boldsymbol{U}_m^T$ and replacing the matrix of eigenvalues, $\boldsymbol{D}_m$, with $\text{diag}(\begin{bmatrix} 1 & 1 & \epsilon \end{bmatrix}^T)$ where ϵ is a small constant (e.g., 0.001) that represents the smallest eigenvalue.

The GMM provides a compressed representation of tens or hundreds of thousands of points in a sensor observation with a small number (several tens or hundreds) of mixture components. Representing the sensor observation in this way yields fast and robust registration because the compactness of this representation enables each pair of mixture components from the source and target distributions to be compared in the registration cost function much more quickly than would be possible when operating directly on the uncompressed point cloud. In contrast, point cloud-based techniques like ICP and GICP must bound the search for matching points in order to remain real-time viable.

3.2.2 Loop Closure

When the current estimated pose j is within a fixed radius r away from a previously visited pose i represented in the factor graph, the poses are considered candidates for loop closure. The estimated pose difference is used to seed the registration between GMMs $\mathscr{G}_i(\boldsymbol{x})$ and $\mathscr{G}_j(\boldsymbol{x})$ associated with poses i and j in the factor graph. After registration, the updated pose θ is used to transform $\mathscr{G}_j(\boldsymbol{x})$ into the frame of $\mathscr{G}_i(\boldsymbol{x})$ by applying the transform $T(\mathscr{G}_j(\boldsymbol{x}), \theta)$. To determine if the GMMs partially overlap the same scene, the Cauchy–Schwarz divergence [17] is used to measure the difference between $T(\mathscr{G}_j(\boldsymbol{x}), \theta)$ and $\mathscr{G}_i(\boldsymbol{x})$. This measure is employed by Kampa et al. [17] as an entropic measure for the classification of distributions and has the same extrema as that of the cost function used in registration. The loop closure problem may be viewed as a classification problem where the goal is to determine whether the scene under consideration by the current view has been previously observed.

Equation (10) is the Cauchy–Schwarz divergence that measures the difference between probability density functions. A closed-form expression for this equation for mixtures of Gaussians is derived in [17].



(a) Aerial system flies in Rapps Cave (b) Aerial system

Fig. 2 **a** A custom-built aerial system collects data in an expansive cavern of Rapps Cave in WV, USA. **b** The aerial system is equipped with a VLP-16 laser scanner

$$D_{CS}\Big(\mathcal{G}_i(\boldsymbol{x}), T(\mathcal{G}_j(\boldsymbol{x}), \theta)\Big) = -\log\left(\frac{\int \mathcal{G}_i(\boldsymbol{x})T(\mathcal{G}_j(\boldsymbol{x}), \theta)d\boldsymbol{x}}{\sqrt{\int \mathcal{G}_i(\boldsymbol{x})^2 d\boldsymbol{x} \int T(\mathcal{G}_j(\boldsymbol{x}), \theta)^2 d\boldsymbol{x}}}\right) \quad (10)$$

This measure is not a metric because it does not satisfy the triangle inequality, but $0 \leq D_{CS}(\mathcal{G}_i(\boldsymbol{x}), T(\mathcal{G}_j(\boldsymbol{x}), \theta)) \leq \infty$ and it satisfies the symmetry property, meaning that $D_{CS}(\mathcal{G}_i(\boldsymbol{x}), \mathcal{G}_j(\boldsymbol{x})) = D_{CS}(\mathcal{G}_j(\boldsymbol{x}), \mathcal{G}_i(\boldsymbol{x}))$. Furthermore, the distributions $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$ are the same only when $D_{CS}(\mathcal{G}_i(\boldsymbol{x}), \mathcal{G}_j(\boldsymbol{x})) = 0$.

If $D_{CS}(\mathcal{G}_i(\boldsymbol{x}), T(\mathcal{G}_j(\boldsymbol{x}), \theta))$ is less than a pre-defined threshold, an edge is added between the poses i and j in the factor graph. Quantifying how different one distribution is from another provides a robust threshold for determining the existence of loop closures because the distribution represents the spread of the samples in the environment. Points sample the 3D world but there is no guarantee that exactly the same point is observed from consecutive observations. Because observations are composed of thousands of points, only distances between nearest points are considered to remain tractable.

4 Results

The proposed approach is compared to the Surfel Mapping (SuMa) approach developed in [2], which is a highly optimized, parallelized implementation for GPU. The GMM formulation is run single-threaded so the timing performance is not one-for-one comparable. 420 LiDAR[1] observations are collected from a ground vehicle in a mine and 450 observations are collected from an aerial system (shown in Fig. 2b) in a cave.

All of the experiments are run on a low-power, embedded Gigabyte Brix with an Intel i7 8550U CPU, four cores (8 hyperthreads), and 32GB of RAM suitable for use on a SWaP-constrained robotic system. The Cauchy–Schwarz divergence loop closure threshold is set to $-\log(1 \times 10^{-6})$. Both SLAM implementations employ the GTSAM framework [7]. 100-component GMMs are used in both experiments and the parameters are kept constant for both experiments.

---

[1] https://velodynelidar.com/vlp-16.html.

Table 1 Timing analysis for the Mine (also shown in the bar graph in Fig. 4f) and Cave (also shown in the bar graph in Fig. 5h) datasets. The Mahalanobis EM variant is approximately 2.25 times faster than the standard EM approach. With the K-Means++ reduction and point filtering, the runtime reduces approximately by a factor of 7

EM Variant	Mine avg. train. time (s)	Cave avg. train. time (s)
Standard EM	1.888	2.365
Mahalanobis EM	0.848	1.054
Mahalanobis EM & KMeans++ Redux	0.2406	0.322

SuMa is originally developed for the Velodyne HDL-64E[2] so the following parameters are updated to work with the Velodyne VLP-16. The data width and height are changed to 500 and 16, respectively. The model width and model height are changed to 500 and 16, respectively. The fields of view up and down are changed to 15 and -15. The map max. angle is changed to 30, sigma angle to 2, and sigma distance to 2. The loop closure distance is also changed to match the setting of the GMM approach.

Two measures are used to evaluate the SLAM results. The Root Mean Square Error (RMSE) as detailed in [25] and the odometric error. The RMSE is defined as the relative pose error at time step i:

$$\boldsymbol{E}_i := \left(\boldsymbol{Q}_i^{-1}\boldsymbol{Q}_{i+1}\right)^{-1}\left(\boldsymbol{S}_i^{-1}\boldsymbol{S}_{i+1}\right) \tag{11}$$

$$RMSE(\boldsymbol{E}_{1:n}) := \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\|trans(\boldsymbol{E}_i)\|^2\right)^{1/2} \tag{12}$$

where $trans(\boldsymbol{E}_i)$ refers to the translational components of the relative pose error $\boldsymbol{E}_i$, the estimated trajectory $\boldsymbol{S}_1, \ldots \boldsymbol{S}_n \in SE(3)$, and the ground truth trajectory $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n \in SE(3)$. The odometric error is computed as the translation and rotation error between frames 1 and j where $j \in [1, n]$:

$$\boldsymbol{E}_j := \left(\boldsymbol{Q}_1^{-1}\boldsymbol{Q}_j\right)^{-1}\left(\boldsymbol{S}_1^{-1}\boldsymbol{S}_j\right) \tag{13}$$

$$OE(\boldsymbol{E}_{1:n}) := \|trans(\boldsymbol{E}_j)\| \tag{14}$$

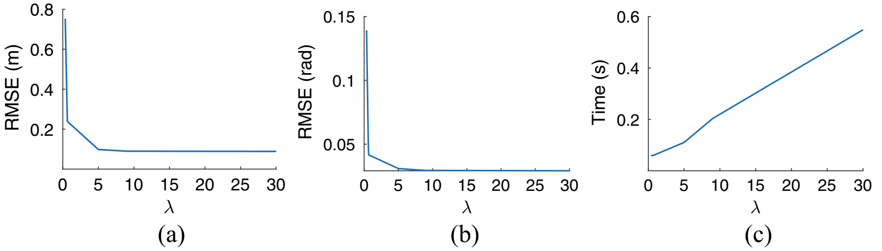
For both relative pose and odometric errors, the rotation errors are similarly computed. Ground truth for the mine dataset is provided by Near Earth Autonomy.[3] GPS is unavailable in the cave so ground truth estimates are obtained using a map generated from a survey-grade, high accuracy FARO scanner.[4]

---

[2] https://velodynelidar.com/hdl-64e.html.

[3] https://www.nearearth.aero/.

[4] https://www.faro.com/.



(a) (b) (c)

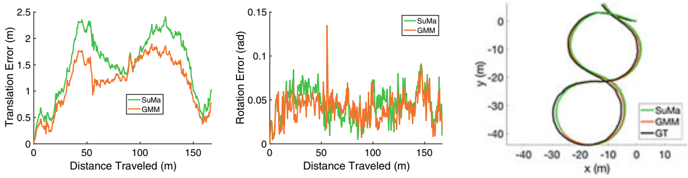
Fig. 3 The λ parameter from Eq. (6) is varied to determine a suitable value that balances accuracy of registration with time to compute the GMM. **a** and **b** demonstrate that for smaller values of λ, the accuracy of the pose estimate decreases, and **c** demonstrates the time to run EM without initialization increases as the value of λ increases. $\lambda = 5$ is chosen to achieve accurate pose estimation while remaining real-time viable

A timing analysis for training the GMM is provided in Table 1. The table illustrates that the Mahalanobis variant of EM from Eq. (6) reduces the timing by a factor of approximately 2.25. The remaining time for the Mahalanobis EM is due to initialization. To decrease the time even further, points outside of a 15 m range are removed for both GMM initialization and training. This is not done for the SuMA approach because it negatively impacts the quality of the pose estimates. The time for GMM initialization is further reduced by using a downsampled set of points (every fifth point) for initialization and then assigning all remaining points to the closest cluster. Processing the data in this way leads to the training times labeled *Mahalanobis EM & KMeans++ Redux* in Table 1, which is approximately 7 times faster than the standard EM. This initialization and the EM approach are used for training GMMs in the mine and cave experiments. Registration times are also provided in Figs. 4e and 5g. The frame-to-frame registration times for SuMa are reported for timing to compare fairly; however, the more accurate frame-to-model SLAM approach is used in all other plots and tables. For both evaluations, the GMM approach requires less data to transmit than the SuMa approach (Figs. 4g and 5i) due to the compactness of the GMM representation.

An analysis is conducted to determine an adequate value for λ using the mine dataset (see Fig. 3). For the following experiments, $\lambda = 5$ as it yields accurate pose estimates while remaining real-time viable.

4.1 Mine

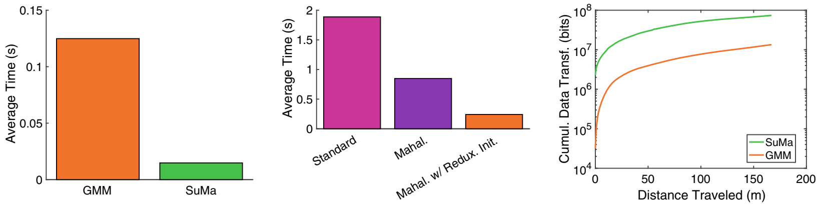
The experiment consists of 420 LiDAR observations in a mine environment taken from a ground vehicle. Figures 4a and b illustrate the error between the estimated pose as a function of distance traveled for the path taken by the vehicle shown in Fig. 4c. Figure 4d presents the RMS errors for consecutive pose estimates as a table. The translation RMSE values are slightly lower for the GMM approach and



(a) Trans. Error (b) Rot. Error (c) Traj. (XY View)

	RMSE (m)
SuMa	9.88×10^{-2}
GMM	9.48×10^{-2}
	RMSE (rad)
SuMa	2.07×10^{-2}
GMM	2.11×10^{-2}

(d) RMSE



(e) Registration Times (f) EM Variant Times (g) Data Transferred

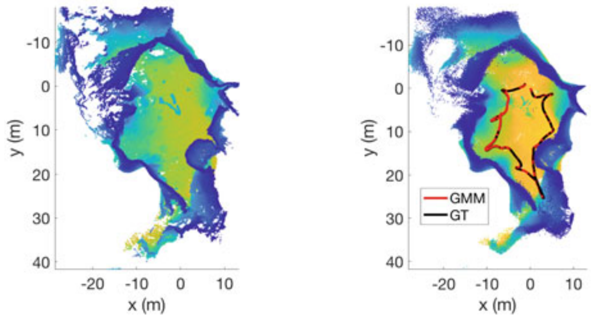
Fig. 4 Results for the Mine dataset. **a** and **b** illustrate the odometric errors as a function of distance traveled. **c** presents a view of the ground truth trajectory in black with each approach overlaid in different colors. **d** presents the RMSE values to evaluate the relative pose error between consecutive sensor observations. **e** presents the registration times and **f** illustrates the timing comparison from the second column of Table 1 as a bar chart. **g** illustrates the cumulative data transferred by each approach as a function of distance traveled

the rotation RMSE values are slightly lower for the SuMa approach. The odometric errors approximately follow the trends of the RMSE values.

The timing results in Figs. 4e and f demonstrate that the Mahalanobis EM with the KMeans++ reduction approach is able to significantly reduce the time to create a GMM. SuMa takes the least time but this approach is parallelized on the integrated GPU in the 8550U.

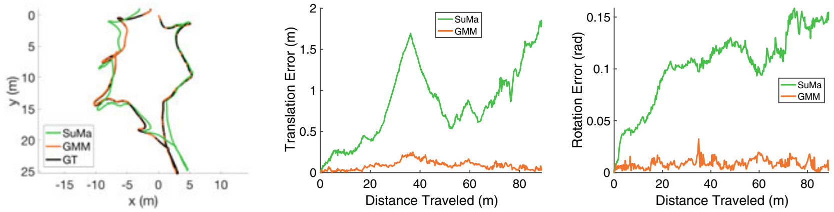
4.2 Cave

Figure 5a illustrates a cross section of the ground truth environment model created from LiDAR observations taken from the aerial system shown in Fig. 2. The path taken by the vehicle is shown in Fig. 5d. In this trial, the GMM approach is able to close the loop between the start and end points of the trajectory which leads to significantly lower overall error. One of the limitations of ICP is the cost function exhibits many local minima that are difficult to overcome when registering point-based sensor observations. Behley and Stachniss [2] attempt to overcome this limitation by trying multiple different initializations for the frame-to-model ICP, but this requires careful tuning of the parameters to ensure that enough variations in the translation and rotation are tested to successfully register the observations. The RMS errors in Fig. 5c

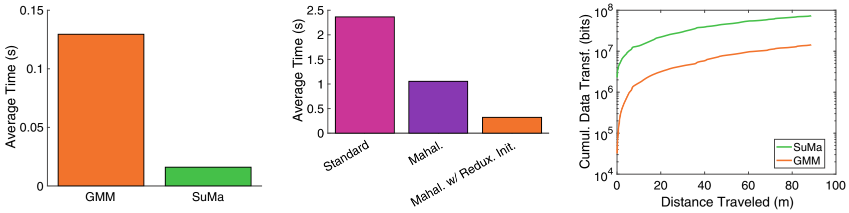


	RMSE (m)
SuMa	3.78×10^{-2}
GMM	2.67×10^{-2}
	RMSE (rad)
SuMa	4.57×10^{-3}
GMM	5.05×10^{-3}

(a) Ground Truth Model (b) GMM Reconstruction (c) RMSE



(d) Trajectory (XY View) (e) Trans. error (f) Rot. error



(g) Registration Times (h) EM Variant Times (i) Data Transferred

Fig. 5 Results for the Cave dataset. (a) and (b) illustrate the ground truth and GMM reconstruction of the environment model. (c) presents the RMSE values for consecutive sensor observations for each approach after running SLAM. (d) presents a view of the ground truth trajectory in black with each approach overlaid in different colors. (e) and (f) illustrate the odometric error as a function of distance traveled. (g) and (h) present a timing comparison for the registration times and EM variants, respectively. (i) illustrates the cumulative data transferred by each approach as a function of distance traveled

demonstrate an overall translation error that is slightly lower for the GMM approach than SuMa. The odometric errors for the trajectory are shown in Figs. 5e and 5f.

Figures 5g and 5h illustrate the almost 7× reduction in training time for the GMM as opposed to the standard EM method. SuMa runs on the integrated GPU in a highly optimized and parallelized way to achieve the reported runtimes. Figure 5b illustrates the GMM reconstruction of the environment by sampling points from the distribution.

5 Discussion and Future Work

While the results for the GMM approach presented in Sect. 4 cannot be run in real-time on a single thread, there is promise for a multi-threaded implementation. The distance calculations in K-Means++, responsibilities in EM, and the correspondence between pairs of mixture components in the GMM registration are all readily parallelizable. Offloading these calculations to a GPU is left as future work to enable real-time performance.

6 Conclusion

This paper demonstrated a real-time viable method for GMM SLAM for compute-constrained systems by formulating an EM algorithm that exploits sparsity to significantly reduce the time to learn a GMM. GMMs are trained from point cloud data, used to estimate pose, and build a map of the environment. A method to close the loop is presented and the approach is evaluated with real-world data of challenging mine and unstructured cave environments.

Acknowledgements The authors would like to thank Aditya Dhawale, Alexander Spitzer, and Kumar Shaurya Shankar for providing feedback on drafts of this manuscript. The authors would also like to thank Curtis Boirum and Brian Osbun for assisting in data collection at Rapps Cave as well as Carroll Bassett of the West Virginia Cave Conservancy for granting access to Rapps Cave.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In : Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035, Jan 2007. https://doi.org/10.1145/1283383.1283494
2. Behley, J., Stachniss, C.: Efficient surfel-based slam using 3d laser range data in urban environments. Robotics: Science and Systems (2018). https://doi.org/10.15607/RSS.2018.XIV.016
3. Bilmes, J.A.: A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Intl. Comput. Sci. Inst. **4**(510), 126 (1998)
4. Bishop, C.: Pattern Recognition and Machine Learning, 2nd edn. Springer-Verlag, New York, New York (2007)
5. Boumal, N., Mishra, B., Absil, P.A., Sepulchre, R.: Manopt, a matlab toolbox for optimization on manifolds. J. Mach. Learn. Res. **15**(1), 1455–1459 (2014)
6. Curtin, R.R.: A dual-tree algorithm for fast k-means clustering with large k. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 300–308 (2017). https://doi.org/10.1137/1.9781611974973.34
7. Dellaert, F.: Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology (2012)
8. Eckart, B., Kim, K., Troccoli, A., Kelly, A., Kautz, J.: Accelerated generative models for 3d point cloud data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5497–5505 (2016). https://doi.org/10.1109/CVPR.2016.593

9. Eckart, B., Kim, K., Kautz, J.: Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 705–721 (2018). https://doi.org/10.1007/978-3-030-01267-0_43
10. Eckart, B.: compact generative models of point cloud data for 3D perception. Ph.D. thesis, Pittsburgh, PA (2017)
11. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proceedings of the 20th international conference on Machine Learning (ICML-03), pp. 147–153, Aug 2003
12. Dimitrios Evangelidis, G., Horaud, R.: Joint alignment of multiple point sets with batch and incremental expectation-maximization. IEEE Trans. Pattern Anal. Mach. Intell. **40**(6), 1397–1410 (2018). https://doi.org/10.1109/TPAMI.2017.2717829
13. Greg Hamerly. Making k-means even faster. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 130–140 (2010). https://doi.org/10.1137/1.9781611972801.12
14. Hosseini, R., Sra S.: An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization. arXiv preprint . arXiv:1706.03267 (2017)
15. Jian, B., Vemuri, B.C.: Robust point set registration using gaussian mixture models. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1633–1645 (2011). https://doi.org/10.1109/TPAMI.2010.223
16. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J.J., Dellaert, F.: isam2: incremental smoothing and mapping using the bayes tree. J. Intl. Robot. Res **31**(2), 216–235 (2012). https://doi.org/10.1177/0278364911430419
17. Kampa K., Hasanbelliu E., Principe, J.C.: Closed-form cauchy-schwarz pdf divergence for mixture of gaussians. In: Proceedings of The 2011 International Joint Conference on Neural Networks, pp. 2578–2585 (2011). https://doi.org/10.1109/IJCNN.2011.6033555
18. Kolouri, S., Rohde G.K., Hoffmann, H.: Sliced wasserstein distance for learning gaussian mixture models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3427–3436, June 2018. https://doi.org/10.1109/CVPR.2018.00361
19. Liu, Y., Nejat, G.: Robotic urban search and rescue: a survey from the control perspective. J. of Intell. Robot. Syst. **72**(2), 147–165 (2013). https://doi.org/10.1007/s10846-013-9822-x
20. O'Reilly, G., Jrad, A., Nagarajan, R., Brown, T., Conrad. Critical infrastructure analysis of telecom for natural disasters. In: Networks 2006. 12th International Telecommunications Network Strategy and Planning Symposium, pp. 1–6, (2006). https://doi.org/10.1109/NETWKS.2006.300396
21. O'Meadhra, C., Tabib, W., Michael, N.: Variable resolution occupancy mapping using gaussian mixture models. IEEE Robot. and Autom. Lett. **4**(2), 2015–2022 (2019). https://doi.org/10.1109/LRA.2018.2889348. Apr
22. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. Robotics: Science and Systems (2009). https://doi.org/10.15607/RSS.2009.V.021
23. Srivastava, S., Michael, N.: Efficient, multifidelity perceptual representations via hierarchical gaussian mixture models. IEEE Trans. Robot. **35**(1), 248–260 (2019). https://doi.org/10.1109/TRO.2018.2878363
24. Stoyanov, T., Magnusson, M., Lilienthal, A.J.: Point set registration through minimization of the l 2 distance between 3d-ndt models. In: 2012 IEEE International Conference on Robotics and Automation, pp. 5196–5201 (2012). https://doi.org/10.1109/ICRA.2012.6224717
25. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2012. https://doi.org/10.1109/IROS.2012.6385773
26. Tabib, W., O'Meadhra, C., Michael, N.: On-manifold gmm registration. IEEE Robot. Autom. Lett. **3**(4), 3805–3812 (2018). https://doi.org/10.1109/LRA.2018.2856279
27. Tabib, W., Goel, K., Yao, J., Dabhi, M., Boirum, C., Michael, N.: Real-time information-theoretic exploration with gaussian mixture model maps. In: Robotics: Science and Systems, FreiburgimBreisgau, Germany, June 2019. https://doi.org/10.15607/RSS.2019.XV.061

Obstacle Climbing of Tracked Robot for Unfixed Cylindrical Obstacle Using Sub-tracks

Ryosuke Yajima and Keiji Nagatani

1 Introduction

When an active volcano erupts, investigations are necessary for disaster prevention and mitigation because of the occurrence of various phenomena that lead to disasters. However, as active volcanoes are dangerous, volcano investigation robots for unmanned investigations are required [3]. Keeping in mind that volcanic environments are rough terrains, although there are various types of robots, tracked robots that have high traversability on rough terrains are suitable for volcanic investigations.

When a tracked robot moves in a volcanic environment, the robot often has to climb over obstacles. Here, obstacles in a volcanic environment are roughly divided into the following two types.

- Obstacles that are fixed on the ground and cannot be moved by the robot (roughness of the ground, buried rocks);
- Obstacles that are not fixed on the ground and can be moved by the robot (loose rocks).

In this paper, the former are called "fixed obstacles" and the latter "unfixed obstacles".

In the field test that our research group conducted at Mt. Aso in 2013, the tracked robot "Quince" had to climb over obstacles. The climbing scene is shown in Fig. 1. Quince slid down due to the movement of obstacles underneath its body. It is necessary to solve such problems for smooth volcano investigations.

There has been considerable research on the obstacle climbing of a tracked robot for fixed obstacles such as steps and stairs [1, 7, 9]. However, although unfixed

---

R. Yajima (✉)
Tohoku University, 468 -1 Aramaki -Aoba, Aoba -ku, Sendai, Miyagi, Japan
e-mail: yajima@frl.mech.tohoku.ac.jp

K. Nagatani
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
e-mail: keiji@ieee.org

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_14

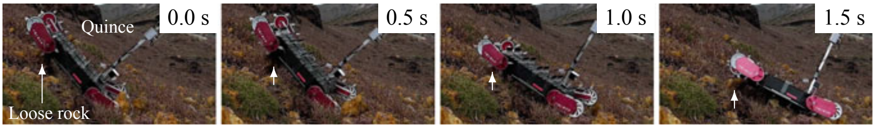


Fig. 1 Tracked robot "Quince" sliding down due to the rolling of unfixed rocks between the robot and the ground. The field is Mt. Aso

obstacles can cause problems leading to failure, there has been little research on unfixed obstacles in the path of tracked robots. Therefore, our research group has been advancing research on unfixed obstacles to understand phenomena that occur when a tracked robot climbs over an unfixed obstacle and to improve climbing performance over unfixed obstacles [10, 11].

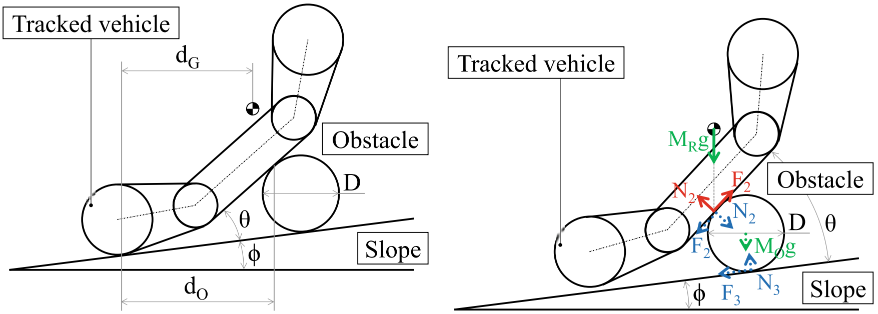
Various multiple-degrees-of-freedom-tracked robots with sub-tracks have been proposed to improve climbing performance [2, 4, 8]. The climbing performance of a robot is improved by suitably controlling sub-tracks [5, 6, 13]. However, it has not been clear whether controlling sub-tracks is a valid solution for unfixed obstacles. In our previous study about the single-tracked robot and unfixed obstacles, although it was suggested that there is a possibility that the robot can easily climb over by changing its attitude, the effect when the attitude of the robot is changed by sub-tracks was not investigated.

In this study, to improve the climbing performance of tracked robots for unfixed obstacles, the effect of sub-tracks on unfixed obstacles is investigated, and motion strategies are considered. In this paper, an unfixed cylindrical obstacle was dealt with as the target obstacle, and conditions that a tracked robot should meet in climbing over the obstacle were derived. Moreover, these conditions and the effect of sub-tracks were verified by an experiment using an actual robot, and motion strategies were considered based on the experimental results.

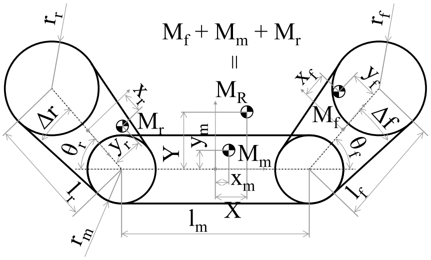
2 Research Scope

In the first step, a simplified environment is considered because an actual volcanic environment is too complex. In this study, it is assumed that a tracked robot climbs directly from the lower side to the higher side of the slope, and the phenomena of the robot are considered in the two-dimensional plane of the robot's side view, as shown in Fig. 2. The target obstacle is an unfixed cylindrical obstacle with a circular cross-section. The slope is flat and solid. Moreover, phenomena are considered from the state that the front end of the main track is on the obstacle, and the approach to the obstacle is not dealt with.

There are roughly three cases for the use of sub-tracks—only front sub-tracks, only rear sub-tracks, and both front and rear sub-tracks. As will be described later, the case that is mainly dealt with is the one with only rear sub-tracks because rear


Tracked vehicle
dG
Obstacle
D
θ
ϕ
Slope
dO
Tracked vehicle
MRg
N2
F2
Obstacle
N2
D
θ
F2
MOg
N3
F3
ϕ
Slope

(a) Model for climbing-over condition 1 (b) Model for climbing-over condition 2


rr
Mf + Mm + Mr
=
MR
xr
Mr
Δr
θr
yr
Y
ym
Mm
xm
X
lr
rm
lm
xf
yf
Mf
Δf
θf
lf
rf

(c) Parameters of the tracked robot with sub-tracks

Fig. 2 Assumed model

sub-tracks largely affect climbing. In addition, to consider the additional effect of front sub-tracks, the case having both front and rear sub-tracks is also considered.

Each symbol is defined as follows.

m, f, r	index that indicates the main track, the front sub-tracks, and the rear sub-tracks
l_m, l_f, l_r	length between the axes of each track
r_m, r_f, r_r	radius of each track
$x_m, y_m, x_f, y_f, x_r, y_r$	centroid position of each track
X, Y	centroid position of the whole robot
M_m, M_f, M_r	mass of each track
M_R	mass of the whole robot
θ	angle between the main track and the slope
θ_f, θ_r	angle between the main track and the sub-tracks
$\varDelta_f, \varDelta_r$	angle between the center line and the track belt of the sub-tracks
D	diameter of the obstacle
M_O	mass of the obstacle
ϕ	slope angle

3 Climbing Over an Unfixed Cylindrical Obstacle Using Sub-tracks

Although this study deals with the case in which only rear sub-tracks are used and the case in which both front and rear sub-tracks are used as described above, the former can be regarded as a special case of the latter, for which each parameter of the front sub-tracks is set to zero. Thus, conditions for the case in which both the front and the rear sub-tracks are used are derived.

3.1 Conditions that a Tracked Robot Should Meet in Climbing Over an Unfixed Cylindrical Obstacle

In the beginning, a sequence of climbing-over by the tracked robot on an unfixed cylindrical obstacle is considered. The sequence is as follows.

1. The front end of the robot climbs the obstacle.
2. The robot moves forward, and the angle θ between the robot and the ground increases gradually.
3. The rear end of the robot leaves the ground when the center of gravity of the robot is just above the contact point between the robot and the obstacle.
4. The robot starts falling forward, and the obstacle is rolled to the higher side of the slope.
5. The front end of the robot contacts the ground.

In this sequence, the important points to climb over are No.3 and No.4. If the center of gravity of the robot does not reach the point just above the contact point between the robot and the obstacle, the robot cannot climb over the obstacle. Furthermore, in No.4, if the obstacle is rolled to a lower side of the slope, the robot slides down and cannot climb over the obstacle.

Based on the above, the conditions that a tracked robot should meet to climb over an unfixed cylindrical obstacle are the following.

Climbing-over condition 1
The center of gravity of the robot reaches the point just above the contact point between the robot and the obstacle.
Climbing-over condition 2
The obstacle is rolled to the higher side of the slope.

When both conditions are satisfied, the robot can climb over the obstacle. In the following sections, each detailed condition is derived, and the effect of sub-tracks on them is considered.

3.1.1 Climbing-Over Condition 1: The Center of Gravity of the Robot Reaches the Point Just Above the Contact Point Between the Robot and the Obstacle

This condition is a geometric condition that is determined by the relative position of the robot and the obstacle. Therefore, it is the same in the case of fixed obstacles and unfixed obstacles, and the condition described in reference [11] can be used. This condition is shown in the following equation by using the distances d_G and d_O, which are shown in Fig. 2a. The distance d_G is from the contact point between the robot and the ground to the center of gravity of the robot. The distance d_O is from the contact point between the robot and the ground to the contact point between the robot and the obstacle.

$$d_g = d_o. \tag{1}$$

The distances d_G, d_O and the centroid position of the robot X, Y are represented as the following equations:

$$d_G = l_r \cos(\theta - \theta_r + \phi) - r_r \sin\phi + \frac{l_m}{2}\cos(\theta + \phi) + \sqrt{X^2 + Y^2}\cos\left(\theta + \phi + \tan^{-1}\frac{Y}{X}\right),$$

$$d_O = \left\{ l_r \cos(\theta - \theta_r) + r_m \sin\theta - \frac{D}{2}(1 + \cos\theta)\tan\phi \right. \\ \left. + \frac{\frac{D}{2}(1 + \cos\theta) - r_r - l_r \sin(\theta - \theta_r) + r_m \cos\theta}{\tan\theta} \right\} \cos\phi,$$

$$X = \left[M_m x_m + M_f \left\{ \frac{l_m}{2} + \left(\frac{l_f}{2} + x_f\right)\cos\theta_f - y_f \sin\theta_f \right\} \right. \\ \left. + M_r \left\{ -\frac{l_m}{2} - \left(\frac{l_r}{2} - x_r\right)\cos\theta_r + y_r \sin\theta_r \right\} \right] / M,$$

$$Y = \left[M_m y_m + M_f \left\{ \left(\frac{l_f}{2} + x_f\right)\sin\theta_f + y_f \cos\theta_f \right\} + M_r \left\{ \left(\frac{l_r}{2} - x_r\right)\sin\theta_r + y_r \cos\theta_r \right\} \right] / M.$$

When Eq. 1 is satisfied, the rear end of the robot leaves the ground and moves to the next step. From reference [11], it is enough just to satisfy this condition to climb over a fixed obstacle.

3.1.2 Climbing-Over Condition 2: The Obstacle Is Rolled to the Higher Side of the Slope

The forces that work at each contact point must be considered for deriving this condition. Here, it is assumed that the velocity of the robot is sufficiently low, the motion is quasi-static, and the condition is considered based on statics. When Eq. 1 is satisfied, the total weight of the robot is supported at the contact point between the robot and the obstacle. The forces that work at each contact point are defined as shown in Fig. 2b. Then, to roll the obstacle to the higher side of the slope, the following equation must be satisfied.

$$F_2 < F_3.$$

The forces that work at each contact point can be derived as the following equations from the equilibrium of the forces:

$$\begin{aligned} N_2 &= M_R \cdot g \cdot \cos(\theta + \phi), & F_2 &= M_R \cdot g \cdot \sin(\theta + \phi), \\ N_3 &= (M_R + M_O) \cdot g \cdot \cos\phi, & F_3 &= -(M_R + M_O) \cdot g \cdot \sin\phi. \end{aligned}$$

By solving these equations, the condition can be represented as the following:

$$\theta < \sin^{-1}\left(-\frac{M_R + M_O}{M_R}\sin\phi\right) - \phi. \tag{2}$$

When Eq. 2 is satisfied, the obstacle is rolled to the higher side of the slope, and the robot climbs over it. In reference [10], which is a study about a single-tracked robot and unfixed obstacles, the sliding-down was dominant. That is because the single-tracked robot could not satisfy this condition.

3.1.3 The Effect of Sub-tracks on the Two Climbing-Over Conditions

The angle θ between the robot and the slope, the contact points of the robot, and the centroid position of the robot are related to the two climbing-over conditions. Here, the effect of sub-tracks on them is considered.

First, in the case with only front sub-tracks, even if the front sub-track angle θ_f is changed, the angle θ is not changed and the contact points are not moved unless the front sub-tracks come in contact with the ground. Although the center of gravity of the whole robot can be moved, it is not by much because the weight of sub-tracks is generally much smaller than the main track. Therefore, it is concluded that the effect of front sub-tracks on the conditions is small.

Secondly, in the case with only rear sub-tracks, when the rear sub-track angle θ_r is changed, the angle θ can be changed because the main track moves accordingly. If it is assumed that the contact point between the robot and the ground is not moved, the contact point between the robot and the obstacle can be moved according to the rear sub-track angle. Moreover, the center of gravity of the whole robot can also be moved. Therefore, the rear sub-tracks affect the conditions and have the possibility of increasing the climbing performance for unfixed obstacles.

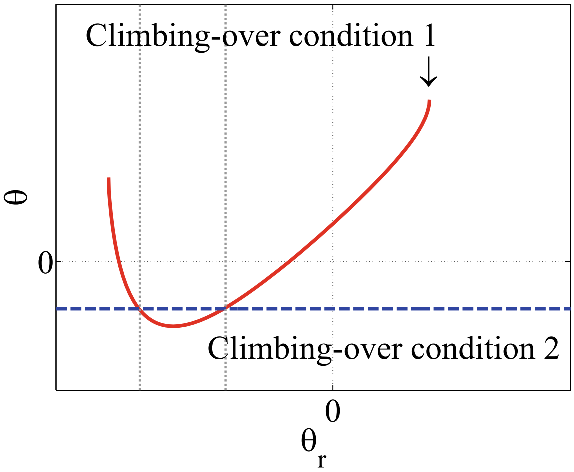
Finally, although the case having both front and rear sub-tracks is almost the same as the case having only rear sub-tracks, the moving amount of the center of gravity slightly increases by the motion of the front sub-tracks.

Based on the above, the rear sub-tracks particularly affect the two climbing-over conditions, and it is considered that the climbing performance of tracked robots for unfixed obstacles can be increased by optimally controlling the rear sub-tracks.

3.1.4 Calculation of Climbing-Over Conditions

Substituting an obstacle diameter D and a slope angle ϕ and solving the climbing conditions in the case with only rear sub-tracks, a graph as shown in Fig. 3 can be obtained. The vertical axis represents the angle θ between the robot and the slope and the horizontal axis represents the rear sub-track angle θ_r. The solid red curve indicates the climbing-over condition 1. On the other hand, the broken blue line indicates the boundary of the climbing-over condition 2 and the area under the broken line satisfies the condition. Therefore, both climbing-over conditions are satisfied in the range of θ_r where the solid red curve is under the broken blue line (between the gray dotted lines). When the robot with a rear sub-track angle in this range climbs the obstacle and the angle θ reaches the value of the solid red curve, the robot climbs over the obstacle. Also, in the range of θ_r where the solid red curve is above the broken blue line, when the angle θ reaches the value of the solid red curve, the robot might slide down.

Fig. 3 Example of calculation of the climbing conditions in the case having only rear sub-tracks



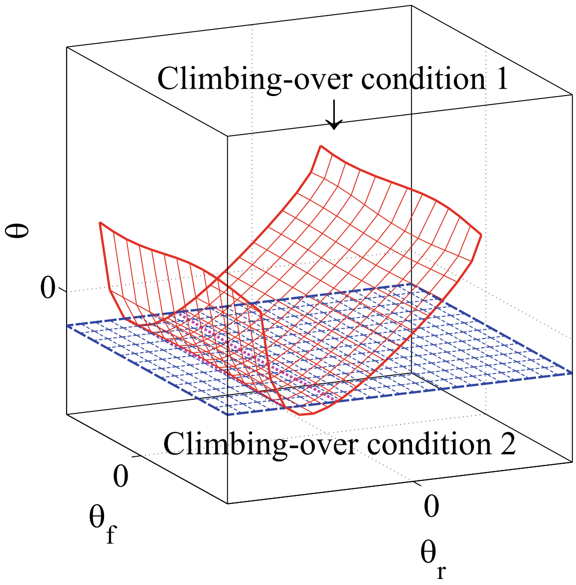


Fig. 4 Example of calculation of the climbing conditions in the case having both front and rear sub-tracks

Moreover, in the case with both front and rear sub-tracks, a three-dimensional graph with an added axis for the front sub-track angle θ_f as shown in Fig. 4 can be obtained. The changes in the direction of the front sub-track angle θ_f are smaller than the changes in the direction of the rear sub-track angle θ_r like the curved surface of Fig. 4, and the effect of the front sub-tracks is small as described above.

4 Experiments

4.1 Description of the Experiment

The purpose of this experiment is to verify the validity of the above climbing-over conditions and to find out whether a robot can climb over an unfixed obstacle by rotating its sub-tracks. In this experiment, a tracked robot with only rear sub-tracks climbed an unfixed cylindrical obstacle. Then, the sub-track angle was changed, and changes in the robot's behavior by changing its attitude were examined.

4.2 Experimental Equipment and Environment

The tracked robot "Kenaf" was used [12]. Kenaf was equipped with only two rear sub-tracks. Kenaf's specification is shown in Table 1. The velocity of the robot was

Table 1 Specification of Kenaf

	Main track		Front sub-track		Rear sub-track	
Mass [kg]	M_m	18.2	M_f	1.6	M_r	1.6
Length [mm]	l_m	470	l_f	155	l_r	155
Radius [mm]	r_m	47.5	r_f	84	r_r	84
Position of center of gravity [mm]	x_m	-13	x_f	16	x_r	-16
	y_m	10	y_f	0	y_r	0
Track angle [°]	–		Δ_f	13.62	Δ_r	13.62

set at a sufficiently low speed: 0.05 m/s. The initial attitude was the state at which the front end of the main track was on the obstacle and the rear sub-track contacted the ground as shown in Fig. 5.

The rear sub-tracks were rotated to a target sub-track angle at a sufficiently high speed in order to reach the target sub-track angle before the center of gravity of the robot reached the point just above the contact point between the robot and the obstacle.

The angle θ between the main track and the slope was measured by the IMU equipped on Kenaf. The sub-track angle θ_f and θ_r were measured by the rotary encoders equipped on the motors of the joints.

As unfixed cylindrical obstacles, polyvinyl chloride pipes with diameters of 115 mm and 166 mm were used. In addition, to prevent slipping, non-slip tape coated with mineral particles were fastened over the pipes.

The experimental simulated slope consisted of aluminum frames, and a plywood board was used as the slope. The incline angle of this slope can be changed to any angle. A urethane sheet with high friction was fastened to the surface of the slope to prevent slipping.

4.3 Climbing-Over Conditions for Kenaf

Using the method in the previous section, the climbing-over conditions for Kenaf were calculated. In the beginning, the boundary whether the attitudes to satisfy both two climbing-over conditions exist, that is, the condition when the solid red curve of climbing-over condition 1 touches the broken blue line of climbing-over condition 2, was calculated. This is shown in Fig. 6. The vertical axis represents the obstacle diameter D and the horizontal axis represents the slope angle ϕ. This graph indicates

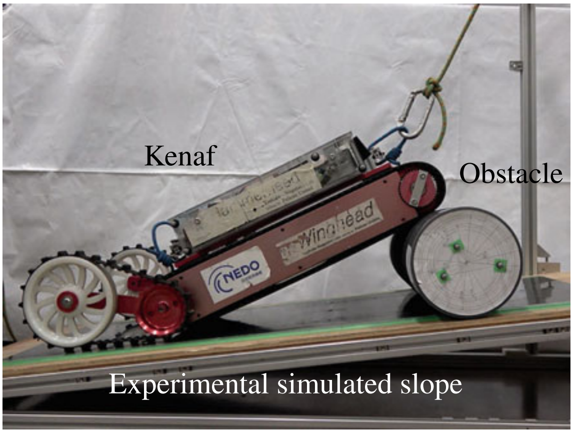


Fig. 5 Experimental equipment

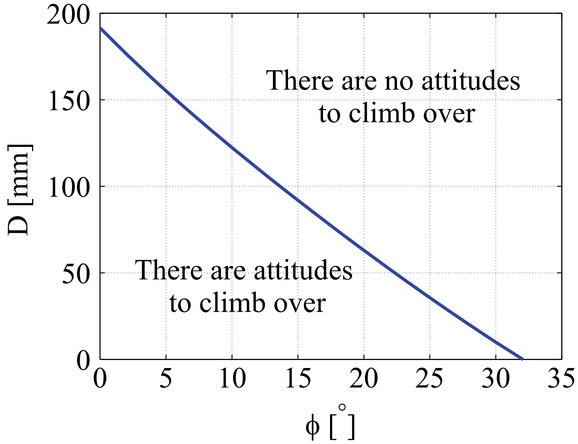
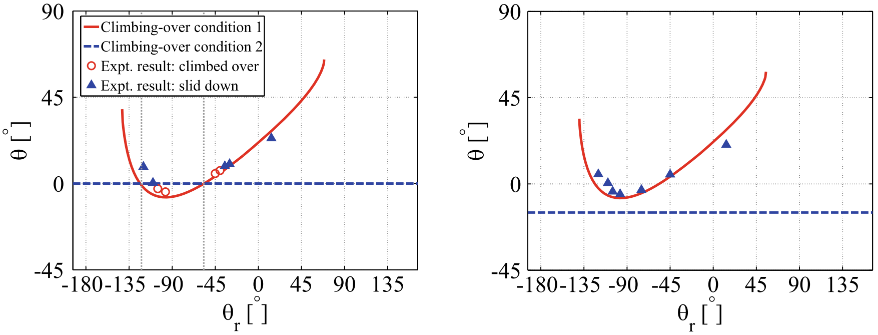


Fig. 6 The boundary whether the attitudes to satisfy both two climbing-over conditions exist

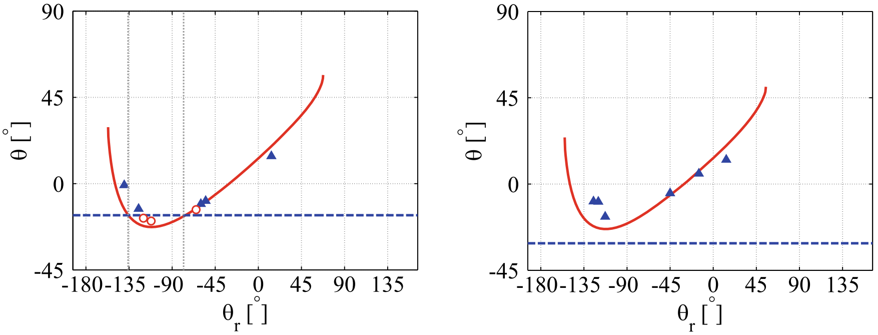
whether the sub-track angles to climb over exist when the tracked robot climbs an obstacle with a diameter of D on a slope with a slope angle of ϕ. There are sub-track angles to climb over in the area under the curve in the graph, and there are no sub-track angles to climb over in the area above the curve.

In this experiment, the following four experimental conditions were verified based on this calculation.

Expt. condition 1
$D = 166$ [mm], $\phi = 0$ [°]
Expt. condition 2
$D = 166$ [mm], $\phi = 7$ [°] (No sub-track angles to climb over)
Expt. condition 3
$D = 115$ [mm], $\phi = 8$ [°]
Expt. condition 4
$D = 115$ [mm], $\phi = 15$ [°] (No sub-track angles to climb over)



(a) Expt. condition 1: $D = 166$ [mm], $\phi = 0$ [°] (b) Expt. condition 2: $D = 166$ [mm], $\phi = 7$ [°]



(c) Expt. condition 3: $D = 115$ [mm], $\phi = 8$ [°] (d) Expt. condition 4: $D = 115$ [mm], $\phi = 15$ [°]

Fig. 7 Experimental results

The graphs of the climbing-over conditions in each experimental condition are shown in Fig. 7. The target rear sub-track angles examined in this experiment depend on the experimental conditions.

4.4 Results

The experimental results are shown in Fig. 7. The angle θ when Kenaf with a target rear sub-track angle θ_r climbed over or slid down in this experiment was plotted in the graph of the climbing-over conditions. The vertical axis represents the angle θ between the robot and the slope and the horizontal axis represents the rear sub-track angle θ_r. The solid red curve and the broken blue line represent the climbing-over conditions 1 and 2, respectively. The red circle denotes the attitude that the robot climbed over the obstacle and the blue triangle denotes the attitude that the robot slid down. If the derived climbing-over conditions are valid, the red circles are plotted on the solid red curve under the broken blue line, and the blue triangles are plotted

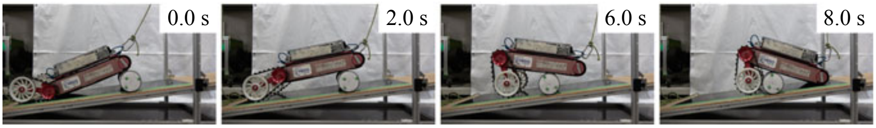


Fig. 8 Motion of Kenaf: $D = 115\,[\mathrm{mm}]$, $\phi = 8\,[°]$, $\theta_r = -111.9\,[°]$, Kenaf climbed over

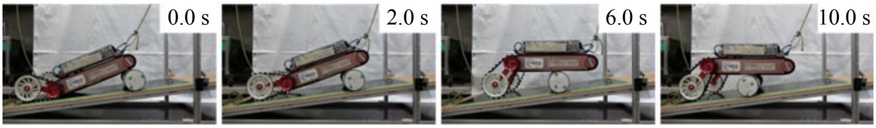


Fig. 9 Motion of Kenaf: $D = 115$ [mm], $\phi = 8\,[°]$, $\theta_r = -55.0\,[°]$, Kenaf slid down

on the solid red curve above the broken blue line. In addition, Figs. 8 and 9 show motion of Kenaf.

In experimental conditions 2 and 4, the robot should not be able to climb over the obstacle with any sub-track angle. The results show that the robot slid down almost according to the derived conditions and could not climb over.

On the other hand, in experimental conditions 1 and 3, sub-track angles to climb over the obstacle exist. The results show that the robot climbed over the obstacle at the sub-track angles to climb over, and slid down at the other sub-track angles almost according to the derived conditions. The reasons for the error at the boundary between climbing-over and sliding-down are (1) measurement error of the centroid position of Kenaf and (2) the existence of grousers on track belts of the sub-tracks.

Moreover, in all experimental conditions, when the sub-track angle is high in minus (left side of the graphs), the results do not follow the climbing-over conditions because the rear sub-tracks touches the obstacle.

From the above experimental results, it is considered that the climbing-over conditions proposed in this paper are valid unless the sub-tracks touch an obstacle. Therefore, depending on the obstacle and the slope, the tracked robot can improve its capability of climbing over obstacles by rotating its sub-tracks to the optimal sub-track angle.

Furthermore, in some cases, Kenaf slipped or slid down before the angle θ reached the value obtained from the climbing-over condition 1. This is because the frictional force became insufficient. The above climbing-over conditions are the conditions when the center of gravity of the robot reaches just above the obstacle without sliding down and do not include the effect of friction and sliding down. To understand this phenomenon, the sliding-down condition considering the friction is necessary apart from the climbing-over conditions.

5 Discussion

5.1 Motion Strategy of Rear Sub-tracks

For a tracked robot to climb over an unfixed obstacle, the rear sub-tracks must be rotated to a sub-track angle at which the robot can climb over the obstacle before the center of gravity of the robot reaches the point just above the contact point between the robot and the obstacle. Thus, the target sub-track angle and the way to rotate the sub-tracks should be considered.

The rear sub-track angle at which the robot climbs over the obstacle has a range. The difference between the angles in the range is the size of the angle θ between the robot and the slope when the robot climbs over the obstacle. The optimal rear sub-track angle depends on the case. If it is desired to reduce the shock to the robot when it climbs over, the rear sub-track angle that the angle θ makes minimum is better. If it is desired to keep the robot and the ground parallel to each other, the rear sub-track angle that the angle θ makes zero is better.

The way to rotate the rear sub-track angle has two policies. The first is to not move the obstacle as much as possible. In the experiment, as the forward speed of the robot and the rotational speed of the rear sub-tracks were constant, the main track was pulled backward, and the obstacle rolled to the lower side of the slope. That is because the rear sub-tracks were rotated. However, the robot may be able to climb over the obstacle without obstacle rolling by cooperative control between the robot's forward movement and the rotation of the rear sub-tracks. The second is to roll the obstacle to the higher side of the slope. The obstacle is rolled to the higher side of the slope when climbing-over occurs, and it is rolled to the lower side of the slope when sliding-down occurs. Thus, when the robot moves the obstacle, rolling it to the higher side of the slope is safer. The reason why the obstacle was rolled to the lower side in the experiment is the rotational direction of the rear sub-tracks. By rotating the rear sub-tracks in reverse, the main track might get pushed forward and the obstacle may be rolled to the higher side.

5.2 The Effect of the Front Sub-tracks and Utilization

As described above, the front sub-tracks do not affect the climbing-over conditions largely because they are lightweight and only slightly move the centroid position of the whole robot. However, if the front sub-tracks come in contact with the ground in front of the obstacle by rotation, the robot can be in a stable state earlier. Particularly, if the main track can leave the obstacle as shown in Fig. 10, the obstacle can be ignored. Therefore, it is desirable that the front sub-tracks contact the ground in front of the obstacle as soon as possible. The above conditions are the conditions when the front sub-tracks do not touch the ground. To analyze the situation and phenomena when they touch the ground, an additional climbing-over condition must be derived.

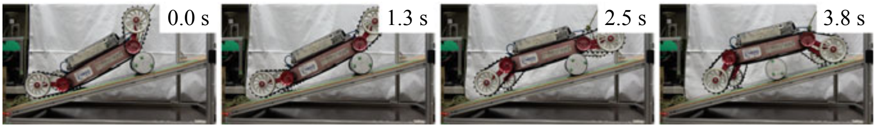


Fig. 10 Climbing over with front and rear sub-tracks by Kenaf

6 Conclusion

In this study, to improve the climbing capability of tracked robots over unfixed obstacles, the climbing-over conditions of a tracked robot with sub-tracks for an unfixed cylindrical obstacle were derived. They were determined from the geometric relationship between the contact points and the rotational direction of the obstacle. The validity of the proposed conditions was verified by the experiment. As a result, the climbing-over conditions are valid, and the tracked robot can climb over the obstacle by rotating its sub-tracks to the optimal sub-track angle depending on the obstacle and the slope. Furthermore, the motion strategies of sub-tracks are discussed based on the experimental results. Although this analysis has a limitation, it is thought that these climbing-over conditions can be used in the real world when a tracked robot climbs an obstacle the cross-section shape of which is close to a circle.

The following are considered for future work: (1) The control described in the motion strategies will be implemented, and the climbing-over motion using sub-tracks will be realized. Moreover, (2) it is necessary to expand to obstacles with more complex shapes and three dimensions. Finally, (3) the robot will recognize obstacles by external sensors mounted on it, determine the target sub-track angle based on the sensor data, and climb over the obstacle automatically.

References

1. Endo, D., Nagatani, K.: Assessment of a tracked vehicle's ability to traverse stairs. ROBOMECH J. **3**(20), 1–13 (2016)
2. Liu, Y., Liu, G.: Track-stair interaction analysis and online tipover prediction for a self-reconfigurable tracked mobile robot climbing stairs. IEEE/ASME Trans. Mechatron. **14**(5), 528–538 (2009)
3. Nagatani, K.: Recent trends and issues of volcanic disaster response with mobile robots. J. Robot Mechatron. **26**(4), 436–441 (2014)
4. Nagatani, K., Noyori, T., Yoshida, K.: Development of multi-D.O.F. tracked vehicle to traverse weak slope and climb up rough slope. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2849–2854 (2013)
5. Ohno, K., Morimura, S., Tadokoro, S., Koyanagi, E., Yoshida, T.: Semi-autonomous control of 6-DOF crawler robot having flippers for getting over unknown-steps. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2559–2560 (2007)
6. Okada, Y., Nagatani, K., Yoshida, K., Tadokoro, S., Yoshida, T., Koyanagi, E.: Shared autonomy system for tracked vehicles on rough terrain based on continuous three-dimensional terrain scanning. J. F. Robot **28**(6), 875–893 (2011)

7. Rajabi, A.H., Soltanzadeh, A.H., Alizadeh, A., Eftekhari, G.: Prediction of obstacle climbing capability for tracked vehicles. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, pp. 128–133 (2011)
8. Rohmer, E., Yoshida, T., Nagatani, K., Tadokoro, S., Koyanagi, E.: Quince: A collaborative mobile robotic platform for rescue robots research and development. In: 5th International Conference on Advance Mechatronics (ICAM2010), pp 225–230 (2010)
9. Tao, W., Ou, Y., Feng, H.: Research on dynamics and stability in the stairs-climbing of a tracked mobile robot. Int. J. Adv. Robot. Syst. **9**, 1–9 (2012)
10. Yajima, R., Nagatani, K.: Investigation of tip-over condition for tracked vehicles climbing over an obstacle on a slope. In: 2017 IEEE/SICE International Symposium on System Integration (SII), pp. 1–6 (2017)
11. Yajima, R., Nagatani, K.: Investigation of the tip-over condition and motion strategy for a tracked vehicle with sub-tracks climbing over an obstacle on a slope. In: 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (2018)
12. Yoshida, T., Koyanagi, E., Tadokoro, S., Yoshida, K., Nagatani, K., Ohno, K., Tsubouchi, T., Maeyama, S., Noda, I., Takizawa, O., Yasushi, H.: A high mobility 6-crawler mobile robot "Kenaf". In: 4th International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (SRMED2007), p. 38 (2007)
13. Zimmermann, K., Zuzanek, P., Reinstein, M., Hlavac, V.: Adaptive traversability of unknown complex terrain with obstacles for mobile robots. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5177–5182 (2014)

Multi-rover Exploration Strategies: Coverage Path Planning with Myopic Sensing

Mickael Laine and Kazuya Yoshida

1 Introduction

1.1 Background and Motivations

The future of planetary exploration is tightly related to multi-rover collaboration and swarm intelligence. This implies that groups of autonomous rovers will work together to complete their mission. Soon, scenes like the one depicted in Fig. 1, could become a mainstream feature of the lunar landscape. With teleoperation difficult for large groups of rovers, not only due to the time delay in communication, autonomous software is essential. Possible missions could be a scientific analysis of vast regions, resource prospecting or scouting locations for future human settlements. In any case, one of the vital systems is an efficient path planning method that has to be tailored depending on rover hardware and mission objectives. Taking lunar surface prospecting as an example, the sensor dedicated to resource detection will most likely require ground sampling and analysis. The path planning algorithm must allow the rover to visit most if not all of the environment. To do so, algorithms known as Coverage Path Planning (CPP) must be used. Relying on GPS and global mapping, it is relatively easy to develop path planning and navigation solutions. In reality, this type of information is not expected to be available in planetary exploration scenarios.

Another point of consideration is related to the internal hardware of the rovers. Both large planetary rovers like NASA's Curiosity and lighter micro-rovers [16], such as the Koguma platform [9] shown in Fig. 2, have severe limitations in computation power. Autonomous rovers require robust software to ensure they function properly

---

M. Laine (✉) · K. Yoshida
Dept. of Aerospace Engineering, Tohoku University, Sendai, Japan
e-mail: mickael@astro.mech.tohoku.ac.jp

K. Yoshida
e-mail: yoshida@astro.mech.tohoku.ac.jp

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_15

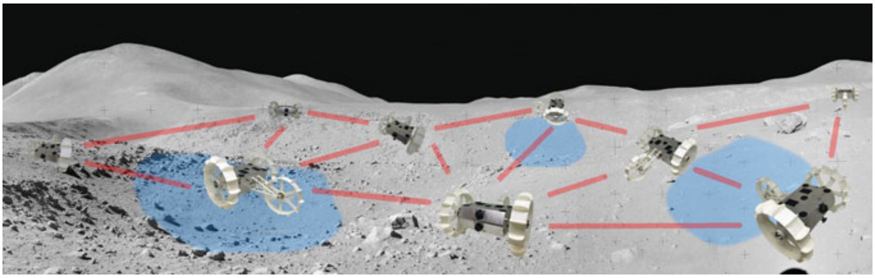


Fig. 1 A representation of inter-connected micro-rovers performing resource prospecting in a lunar scenery

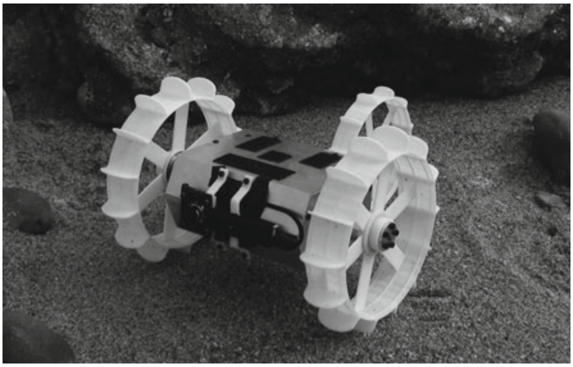


Fig. 2 Koguma, a micro-rover prototype for future lunar exploration missions designed and built at Tohoku University. This prototype is one of the smallest and lightest ever created weighing only 1 kg

throughout their lifetime and recover from potential failures. Probably the most important system is Simultaneous Localization and Mapping (SLAM). Knowing the location of the rover for navigation and mission coordination is important, but when prospecting for resources, having accurate locations of ground sampling is essential. However, SLAM is computationally demanding, as seen in [15], where they benchmark the requirements and efficiency of some well-known algorithms. Adding the needs of SLAM and all other necessary systems, the path planning solution needs to be able to work in real time with minimum computation requirements. Micro-rovers are also limited by their sensing capabilities and can only perceive their surroundings within a limited range, as seen in Fig. 3. Developing path planning solutions relying solely on this myopic information is challenging and has currently not been well studied.

This paper presents an approach toward multi-rover coverage for the purpose of lunar surface resource prospecting. Considering the hardware limitations of current planetary exploration rovers, we focus on building a real-time exploration algorithm for lunar surface coverage. Continuing with the aspect of limited rover capabilities, we also take into account the range limitations of the sensors by choosing a myopic approach. The rovers can only perceive the environment within a short range of their current position and decide their next actions depending on this information. These

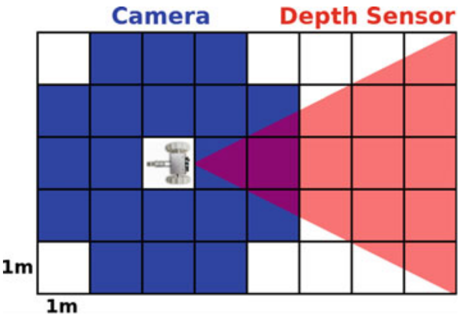


Fig. 3 A visual representation of myopic sensing information as could be expected from a lunar exploration micro-rover

properties, along with the basic building blocks of the algorithm are explained in Sect. 2. In Sect. 3 we compare coverage results, first in the case of single rover autonomy, then for multi-rover systems. Finally, Sect. 4 comments on the presented results and proposes the next steps to increase the efficiency of the algorithms. Although this paper does not present results from field testing, the hardware limitations and capabilities of the Koguma platform were taken into consideration during the development of the algorithms.

1.2 Related Work

The development of path planning strategies for planetary exploration comes with a most unique combination of constraints and difficulties to overcome. The soft soil covering all of the planetary surfaces, associated with a large number of unpredictable natural obstacles, make it difficult to achieve accurate planning over large distances. A mission applicable exploration algorithm will have to take into account environmental conditions, limited sensor capabilities and be capable of on-the-fly replanning. In most current research for planetary exploration [14], the path planning algorithm consists of a graph-based method that minimizes motion cost between a predefined start and goal position. The cost function used to give weight to the edges of the graph is defined as a combination of actual motion capabilities, terramechanical considerations and power consumption. The terramechanical factors are obtained experimentally and represent the capability of the rover to move in the lunar regolith. This is a good example of smart cost allocation based on environmental factors and rover mobility. The minimal cost path is finally obtained by applying Dijkstra's algorithm to the weighted map. In [12], the algorithm uses digital elevation maps and active visual-based terrain classification to understand the surrounding environment (within a limited range) and generate its cost function. A final example in this line of exploration strategies is given in [13], where the variant this time is that the environment is perceived using a LIDAR, then segmented into a triangular mesh pattern that defines the graph and finally the path is obtained with A*. The main conclusions from this are that usual algorithms should have to deal with limited sensing range and adaptability to both unplanned objects and environmental complexity.

Coverage Path Planning (CPP) includes a wide range of algorithms that aim to explore an environment in its entirety. Basic knowledge of this subject can be found in [1, 4, 7], where the authors have conducted surveys on a widespread of methods and applications. An important aspect of CPP is environmental decomposition. Depending on the goal of the exploration method, the environment must be segmented into regions of various sizes and shapes. A path generation method is then used to explore all of these regions. One main aspect that makes these surveys notable is a listing of current coverage methods with their intended applications, advantages and shortcomings.

A popular approach to CPP is a graph-based algorithm known as Spanning Tree Coverage (STC). In essence, STC computes a tree, spanning from a starting position for the rover and that branches out to every cell in the environment. Following this, each cell is subdivided into smaller sub-cells and the rover will circumnavigate the tree until it returns to its starting position. The three basic variants of STC (off-line, on-line, and ant-like) are well explained in [2]. There are also variants of STC, called Spiral-STC and Scan-STC given in [3]. These two algorithms differ from standard STC by the basic rover motion scheme, where one uses a spiral-like motion and the other a Boustrophedon pattern. They both also introduce a new system that allows for cell revisitation in order to properly explore the entirety of the exploration region. In [5], STC is applied to multi-rover CPP using a two-step algorithm. The first step generates a single STC solution that covers the entirety of the grid. The solution is then divided among the different rovers who execute the exploration task together. Finally, a different approach to multi-rover coverage is given in [6]. This algorithm takes into account a known environment and rover initial positions to divide the exploration region equally among the rovers then applies STC individually for each member. To summarize, graph-based approaches and, in particular, STC represents one branch of CPP but present some issues. These graph-based approaches require prior knowledge of the environment and in particular, the location of obstacles. We present a path planning solution for environment coverage that does not require these high-resolution and accurate maps but instead, relies only on limited onboard sensor information.

For this paper, we need to define a simpler approach to coverage that supports unknown environments and can be expanded to multiple rovers operating in the same space. One such example is given in [11], where they use basic motion schemes and compare their efficiency. The objective is to map the edges of the environment and contours of obstacles. Although this is quite different from planetary exploration, the results obtained from each basic motion can be used and adapted to our needs. Another approach for coverage of unknown environments is presented in [8], which does not focus on the motion scheme itself but instead, applies a simple Boustrophedon pattern and proposes an efficient backtracking method. Most CPP algorithms try to limit, if not suppress completely, backtracking as it is contrary to their definition of efficiency (i.e. shortest path and single-cell visitation). The solution presented uses overall map geometry and memory of previously sensed, but unexplored cells to minimize total path length and ensure complete coverage. Although we also use similar basic motion patterns and allow for backtracking, we present a method that does not store or use accumulated data over time. Instead, we compute single-step planning based on current sensor information and the exploration path up to that point. Finally, an interesting concept towards solving the issue of multiple rovers in the same workspace is explained in [10]. Although it is fairly older than other related works introduced here, the algorithm utilizes myopic sensing, predefined decision-based path planning and solutions for multi-rover coordination in unknown environments without communication between the rovers. What is most notable is the strict assumptions of fully decentralized computing for rovers with limited capabilities as these are also the foundations of our work.

2 Multi-rover Myopic Exploration

The main objective of this work is to provide a multi-rover path planning solution for resource prospecting on the lunar surface. In essence, the problem is equivalent to finding a CPP solution that satisfies the constraints of such a mission. The algorithm will need to be able to navigate completely unknown environments with multiple agents operating simultaneously and while keeping the computation requirements as low as possible. To achieve this, we have built a one-step, real-time path planning algorithm from its most basic form until it can systematically cover the exploration region in its entirety.

2.1 Problem Formulation and Initial Assumptions

Developing path planning solutions for planetary exploration is already challenging. The rovers need to be fully autonomous and be capable of dealing with a completely unknown environment. Combined with the use of micro-rovers, it becomes even more complex. This category of rovers, generally less than 10 kg, are limited by their computation power and sensing range (generally 10 m range or less). With all the other system requirements such as localization, mapping, and general operation software, the path planning algorithm needs to be as lightweight as possible. Additionally, the sensing capabilities can be defined as myopic, this means that the rover can only perceive its surroundings within a limited range. With this in mind, the approach we have selected is a single-step ahead path planning algorithm that only takes into account the immediate surroundings of the rover to compute the next motion. Based solely on this, we aim to achieve complete environment coverage for single, and multi-rover systems.

For simplicity of development and evaluation in this initial stage, the environment is represented as a 2D grid. It is an exact cellular decomposition with equal-sized cells and a square shape. The exact dimensions of the rovers or cells are not constraints specifically defined in this work, instead they should be considered scalable depending on the hardware available and mission parameters. In the following discussion, it is reasonable to assume that the cells represent an area of 1 m^2 and they can fit a single rover at any given time. Initially, the environment is composed of free cells and a number of randomly distributed obstacles.

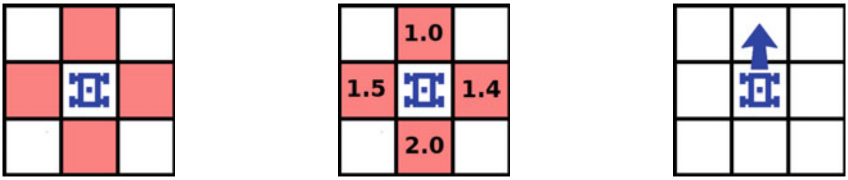
Concerning the rovers, we assume they are able to localize themselves within the environment perfectly. As specified earlier, they have myopic sensing capabilities, and can acquire information omni-directionally limited to a single cell ahead. Their initial knowledge is exclusively the dimensions of the environment, more specifically, they do not know the amount or location of the obstacles. In the case of multi-rover systems, we assume that it consists of a group of homogeneous rovers. Finally, there are no data or distance limitations on inter-rover communications.

2.2 Single Rover Myopic Coverage

In this section, we describe the exploration algorithm as well as the process of developing a minimal cost function. As stated before, the main objective is to achieve complete coverage of an unknown environment using real-time path planning. The basic steps, illustrated in Fig. 4, are to acquire sensor data, compute motion cost, and move to the minimal cost cell. These steps are also presented in Algorithm 1 with a more comprehensive overview of the entire process. The expected outputs of this are the traveled path of the rover as well as a complete map of the environment. The exploration ends once all *Free* cells have been visited.

The motion cost function is developed from the ground up to obtain the minimally sufficient algorithm that guarantees complete coverage. The first version is based solely on a static cost value as shown in Eq. 1. Based on the current orientation of the rover, the cost function takes the values shown in Fig. 4b.

$$c(x, y) = C_{Static} \tag{1}$$



(a) Step 1 - S_{vec} = getSensorInput(p_i)).

(b) Step 2 - C_{vec} = computeCost($p, cell$).

(c) Step 3 - p_{i+1} = MinCost(C_{vec}).

Fig. 4 Visual representation of the fundamental steps associated with the single-step path planning process with myopic sensing

Algorithm 1 Single rover myopic coverage

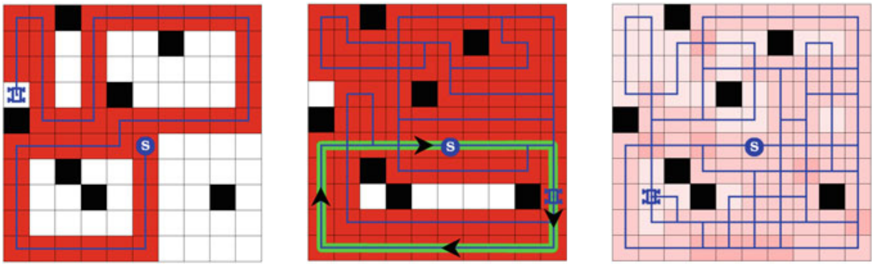
```
Input: Environment size $(X, Y)$, starting pose $p_0$
Output: Traveled path: $Path$, complete environment map $\mathfrak{M}$
$\mathfrak{M} \leftarrow$ initialiaseEnvironment($X, Y$)
$n$= total number of cells
$p_0 = (x_0, y_0, c_0)$
while True do
    $S_{vec} \leftarrow$ getSensorInput($p_i$)
    for each $cell \in S$ do
        if $cell$ is $Obstacle$ then
            $O+ = 1$
        else
            $C_{vec} \leftarrow$ computeCost($p, cell$)
        end if
    end for
    $p_{i+1} \leftarrow$ MinCost($C_{vec}$)
    append $p_i$ to $Path$
    updateMap($\mathfrak{M}$)
    for each $cell \in \mathfrak{M}$ do
        if $cell$ is $Visited$ then
            $V+ = 1$
        end if
    end for
    if $V \geq (n - O)$ then
        Stop exploration
    return $Path$
    else $V = 0$
    end if
end while
```

The resulting motion forms an inward orientated spiral and only authorizes single-cell visitation. This version of the algorithm is not capable of achieving coverage as it always results in the rover entering a dead-end condition, see Fig. 5a. Without the option to backtrack or revisit cells the rover finds itself stuck. To cope with it, we add a second component to the cost function as presented in Eq. 2. The cost named $C_{Visited}$ is a preset penalty that allows the rover to backtrack on previously visited cells while promoting unexplored cells.

$$c(x, y) = C_{Static} + C_{Visited} \tag{2}$$

The result of this method is illustrated by Fig. 5b. In most cases, and due to the deterministic nature of the choices made by the rover, it enters an infinite loop. Once again, this algorithm cannot achieve complete coverage. As a final measure to improve the cost function, we introduce the revisitation factor V_i as seen in Eq. 3. This value represents the number of times the rover visited a specific cell.

$$c(x, y) = C_{Static} + C_{Visited} V_i \tag{3}$$



(a) C_{Static} (b) $C_{Static} + C_{Visited}$ (c) $C_{Static} + C_{Visited} V_i$

Fig. 5 Results obtained while building the minimal cost function that can achieve complete coverage. **a** The rover enters a dead-end scenario and cannot continue. **b** The rover is stuck in a loop represented by the green path. **c** The final configuration that can consistently achieve complete coverage

In Fig. 5c, we can see the environment is completely covered. The hue of the cells gets darker the more they are visited. The increasing penalty that comes with more revisitations acts similarly to a potential function. The rover is pushed away from frequently visited areas of the map until it can find all free cells. This is the simplest cost function that can always achieve complete coverage.

The computational complexity of a CPP algorithm is proportional to the size n of the map. To find a complete solution for environment coverage, the algorithm needs to perform a number n_I of iterations. Each iteration, we need to go through all cells of the map to determine how many cells have been visited. Although it is dependent on the environment composition, the value of n_I is proportional to the number of cells n. This means that the computational complexity for complete coverage is $O(n^2)$, which contradicts the goal of simplicity that motivates this work. However, the fundamental principle of this algorithm is real-time path planning. This implies that the relevant computational complexity is not related to finding the complete solution but that of a single iteration. In that regard, the computational cost to consider is $O(n)$ which is the process of going through the map to count the number of visited cells.

2.3 Extension to Multi-rover Systems

After defining a minimal CPP algorithm for single rovers, we can now look into ways of having multiple rovers collaborate. Each rover implements the algorithm described in the previous section and two multi-rover methods are presented here.

Collaborative exploration: As a natural evolution from the single rover implementation, a group of rovers using the myopic sensing CPP algorithm explore the environment. The rovers are free to travel anywhere on the map and collaboratively conduct coverage. At each iteration, the rovers share their position with the other members so that they constantly all have the same information about the environment. This combined knowledge should allow for faster exploration and the rovers should easily be able to converge to unexplored areas and cells. Until the coverage task is finished, exploration for all members continues and the path length is the same for all rovers, which corresponds to the total exploration time.

Evaluating the efficiency of the collaborative exploration CPP method is good but comparing it to another exploration strategy makes this study even more interesting. With the constraints we defined for myopic sensing micro-rovers, there are few to no existing algorithms that can give significant information about efficiency. Taking inspiration from optimization properties in multi-rover graph-based algorithms. We introduce a second exploration method that uses partitioned environments.

Environment partitioning: Based on the initial position of the rovers, we generate a Voronoï diagram. By computing the Euclidean distance from the initial positions, each cell is assigned to the rover, which minimizes this distance. The rovers then use the myopic CPP algorithm to individually explore their assigned regions. Once each rover completes coverage of their areas, they stop. This means that exploration time is the path length of the final rover to complete coverage of his assigned region. For later development, this simplifies the problem by eliminating overlapping and potential collisions of the rovers.

3 Simulations and Results

In this section, we present simulation results for the algorithms introduced in the previous section. The simulations are conducted in environments of different sizes. The number of obstacles is set to 5% of the total number of cells and their distribution is randomly generated. Each specific test is run in a number of random maps and for each map, we run simulations in a number of random initial positions. To evaluate the efficiency of each algorithm, we introduce the path length ration in Eq. 4.

$$Path\ length\ ratio = \frac{Total\ path\ length}{Total\ number\ of\ cells} \tag{4}$$

This ratio represents the efficiency of the path planning algorithm when compared to a single-pass coverage, which is the best possible result for CPP. For a single rover exploration, a path length ration value of 1.0 is a single-pass solution, whereas a ratio of 2.0 means that on average, every cell has been visited twice.

3.1 Single Rover Coverage Results

In this first set of simulations, we analyze the efficiency of the single rover CPP algorithm in different sizes of environments. The results of which are presented in Table 1. The best results in all cases are close to what would be obtained by a single-

Table 1 Single rover myopic coverage results in maps of different sizes

Environment size	Min path length	Max path length	Average path length	Average path length ratio
10 × 10	97	194	129	1.36
15 × 15	216	465	298	1.39
20 × 20	420	1017	572	1.50
30 × 30	907	1933	1349	1.58

pass algorithm with a low mount of cell revisitations. On the other hand, the worst results show a path length ratio of over 2.0 in all environment sizes. The overall averages, ranging from 1.36 to 1.58, increase along with environment size.

The main issue we observed is that exploration is more or less optimal up until a certain point. Once there are only a few cells left to explore, the algorithm can take time to find them. Apart from cell revisitation, there is no active use of memory in exploration. This means that we have to rely solely on the natural effect of the algorithm to be pushed away from highly visited areas. It is difficult to predict how fast the algorithm will converge to these free cells, which leads to significant variations in the results.

3.2 Reduced Percentage of Completeness

In an attempt to reduce exploration time and the standard deviation of results, we introduce the concept of percentage of completeness. Instead of aiming for complete coverage of the environment, we explore until a certain percent. In an environment of 20 × 20 cells, we run a series of simulations while gradually reducing the required percentage of completeness. We show the average path length ratios and the standard deviation for each percentage as can be seen in Fig. 6.

Both the average path length and the deviation show a steady decrease in values that seem to stabilize the more we reduce the percentage of completeness. By reducing to 95% the amount of coverage, we can observe approximately a 15% decrease in average path length. Especially in the context of planetary applications, reducing exploration time is primordial. When comparing the loss of information to the gain in exploration time, reducing the percentage of required completeness seems like a reasonable compromise. In the next part of this study, we consider exploration finished when we reach 95% of total coverage.

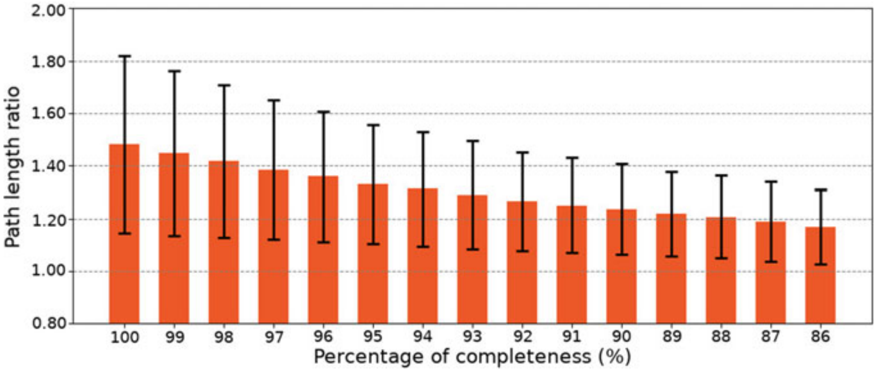
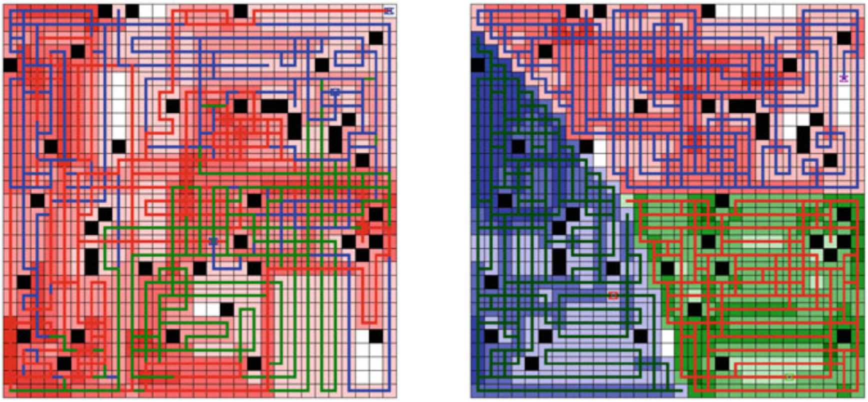


Fig. 6 Comparison of efficiency using the single rover algorithm for various percentages of completeness. Results are based on a large number of simulations (3000 runs for each percentage of completeness) in multiple environments with random starting positions every time



(a) Collaborative exploration method. (b) Partitioned environment method.

Fig. 7 Representation of the final state for the two multi-rover exploration algorithms introduced in this work

3.3 Multi-rover Coverage Results

For the comparison of multi-rover algorithms, we still run simulations on different exploration maps and in each of them, use a number of randomly selected starting positions. Illustrations of the final exploration maps for both methods are presented in Fig. 7. For different sizes of environments, we deploy 3, 4, and 5 rovers and compare the results, which are shown in Fig. 8. When comparing the single and multi-rover results, in the 30 × 30 map, the average path length ratio is not inversely proportional to the number of rovers. There is though a considerable decrease in total path length.

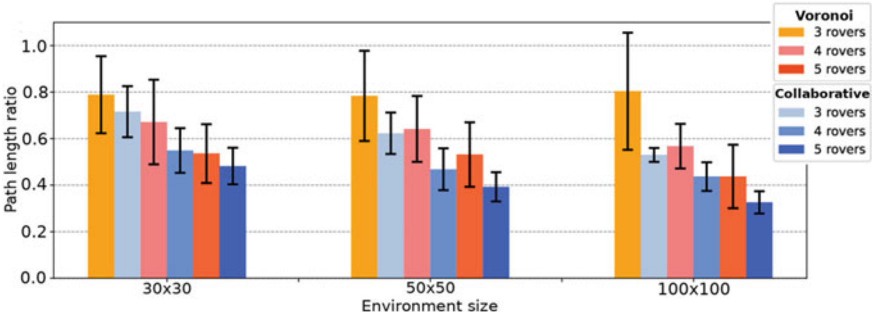


Fig. 8 Results of multi-rover coverage using collaborative exploration and Voronoï partitioning methods

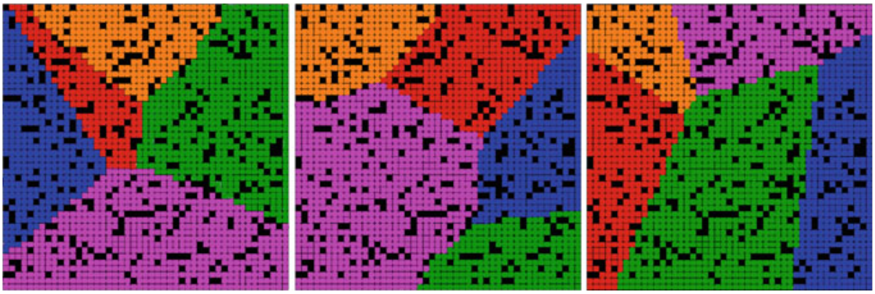


Fig. 9 Voronoï partitioning of the environment based on the initial position of the rovers. This illustrates the issue with a standard Voronoï diagram in that the partitions are not equal among the rovers leading in non-optimal exploration results

For all cases, the data shows faster results when increasing the number of rovers. Depending on the size of the environment, there might be a limit number of rovers over which the exploration time does not or marginally decreases. This should be the focus of a future study and consider more rovers and larger environments as well.

Both multi-rover algorithms show improvements in exploration time compared to a single rover method. The direct comparison of the two demonstrates better results in all cases for the collaborative exploration. As we increase the environment size, the collaborative method improves for all numbers of rovers. In all sizes of environments, the partitioned approach has very small variations in path length values. This is mainly due to the partitioning method used. In Fig. 9, we have the Voronoï partitions that were generated for five rovers when changing their initial positions. As we can see, none of the rovers are assigned an equal exploration area. Since the final path length corresponds to the time the last rover completes exploration, it is proportional to the area size assigned to each rover. In order to obtain closer results between both algorithms probably requires a modification of the partitioning method to obtain equally sized areas.

4 Conclusions and Future Developments

In this paper, we have introduced an approach towards a minimally complex solution to complete coverage path planning for single and multi-rover systems. For the single rover exploration algorithm, the results on total path length can vary widely with the best values nearly equal to single-pass solutions. On average though, we should expect a path length ratio of about 1.45. This result can be considerably improved when slightly lowering the percentage of coverage. By reducing coverage to 95% completeness, we can obtain, on average, a 15% reduction on total path length. This does entail a small loss in gained information about the environment but in the case of planetary exploration systems, minimizing the path length is of higher priority. In the case of multi-rover algorithms, we introduced two possible approaches. The first being a completely collaborative method, where all rovers have full access to the exploration region, and a second that uses Voronoï partitioning to divide the area among the rovers. In the context of this study, the collaborative approach systematically outperforms the partitioned approach in all environment sizes and for any number of tested rovers. We found that this was mainly due to an uneven division of the area from the standard Voronoï algorithm based on Euclidean distance.

In future work, we want to expand these algorithms to larger scale exploration with more realistic lunar mission scenarios. In addition to this, we want to rework the partitioning method in order to equally divide the exploration areas among the rovers. Finally, we want to add some form of higher-level information and processing in order to optimize the results of our coverage path planning.

References

1. Choset, H.: Coverage for robotics-A survey of recent results. A. Math. Artif. Intell. **31**(1–4), 113–126 (2001)
2. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math. Artif. Intell. **31**(1–4), 77–98 (2001)
3. Gabriely, Y., Rimon, E.: Competitive on-line coverage of grid environments by a mobile robot. Computat. Geom. **24**(3), 197–224 (2003)
4. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
5. Hazon, N., Kaminka, G.A.: Redundancy, efficiency and robustness in multi-robot coverage. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 735–741 (2005)
6. Kapoutsis, A.C., Chatzichristofis, S.A., Kosmatopoulos, E.B.: DARP: divide areas algorithm for optimal multi-robot coverage path plannings. J. Intell. Robot. Syst. **86**(3–4), 663–680 (2017)
7. Khan, A., Noreen, I., Habib, Z.: On complete coverage path planning algorithms for non-holonomic mobile robots: survey and challenges. J. Inf. Sci. Eng. **33**(1), 1258–1276 (2017)
8. Khan, A., Noreen, I., Ryu, H., Doh, N.L., Habib, Z.: Online complete coverage path planning using two-way proximity search. Intell. Serv. Robot. **10**(3), 229–240 (2017)

9. Laîné, M., Tamakoshi, C., Touboulic, M., Walker, J., Yoshida, K.: Initial design characteristics, testing and performance optimisation for a lunar exploration micro-rover prototype. Adv. Astronaut. Sci. Technol. **1**(1), 111–117 (2018)
10. Lumelsky, V.J., Harinarayan, K.R.: Decentralized motion planning for multiple mobile robots: the cocktail party model. Auton. Robots **4**(1), 121–135 (1997)
11. Masehian, E., Jannati, M., Hekmatfar, T.: Cooperative mapping of unknown environments by multiple heterogeneous mobile robots with limited sensing. Robot. Auton. Syst. **87**, 188–218 (2017)
12. Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J.: Risk-aware planetary rover operation: Autonomous terrain classification and path planning. IEEE Aerosp. Confer. **1–10** (2015)
13. Rekleitis, I., Bedwani, J.L., Dupuis, E.: Autonomous planetary exploration using LIDAR data. In: IEEE International Conference on Robotics and Automation, pp. 3025–3030 (2009)
14. Sutoh, M., Otsuki, M., Wakabayashi, S., Hoshino, T., Hashimoto, T.: The right path: comprehensive path planning for lunar exploration rovers. IEEE Robot. Autom. Mag. **22**(1), 22–33 (2015)
15. Tuna, G., Gulez, K., Gungor, C.V., Mumcu, V.T.: Evaluations of different simultaneous localization and mapping (SLAM) algorithms. In: IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, pp. 2693–2698 (2012)
16. Walker, J.: Final configuration of the ispace HAKUTO rover for a Google Lunar XPRIZE mission. In: 68th International Astronautical Congress (2017)

Machine Learning Techniques for AUV Side-Scan Sonar Data Feature Extraction as Applied to Intelligent Search for Underwater Archaeological Sites

Nandeeka Nayak, Makoto Nara, Timmy Gambin, Zoë Wood, and Christopher M. Clark

1 Introduction

Locating wrecks and sites of interest to marine archaeologists are challenging and time-consuming, since fine detail and extensive experience is required to pick out wrecks from seafloor features. Archaeologists use indicators like corners, edges, and the scuffing of the surrounding area to differentiate areas of interest from sand, rocks, and other natural terrain features. However, due to the high volume of data, the identification of possible sites is time-intensive and inexact. Different experts will flag different objects as wrecks and occasionally contradict one another.

In order to obtain high-resolution images of the seafloor, marine archaeologists often use side-scan sonar (SSS) (Fig. 1). Often, SSS is used either on a towfish—where the SSS is mounted on a hydrodynamic block towed behind a boat—or, more recently, on autonomous underwater vehicles (AUVs). Once the SSS data is collected, it is converted to images and inspected for regions of interest. If a location seems to be of sufficient interest, archaeologists then revisit the site via either divers, remotely

---

N. Nayak (✉) · M. Nara · C. M. Clark
Harvey Mudd College, Claremont, CA, USA
e-mail: nnayak@hmc.edu

M. Nara
e-mail: mnara@hmc.edu

C. M. Clark
e-mail: clark@hmc.edu

T. Gambin
University of Malta, Msida MSD 2080, Malta
e-mail: timmy.gambin@um.edu.mt

Z. Wood
California Polytechnic State University, San Luis Obispo, CA, USA
e-mail: zwood@csc.calpoly.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_16

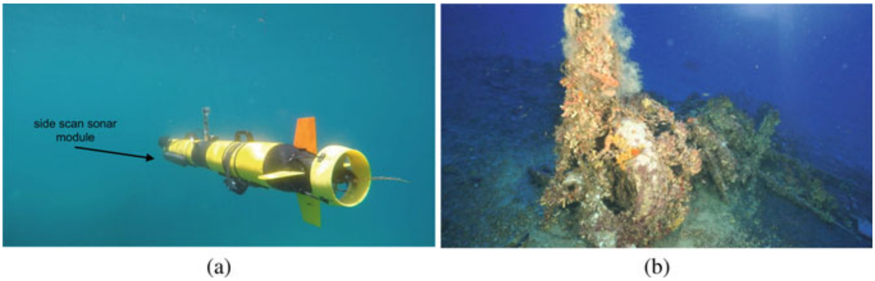


Fig. 1 **(a)** Iver3 AUV with SSS at the front and a GoPro HERO4 attached under the rear handle. **(b)** Remains of the World War II-era Fairey Swordfish discovered off the coast of Malta

operated vehicles (ROVs), or AUVs, depending on the site's accessibility. If this results in a serious discovery, the site is often mapped with photogrammetry and a three-dimensional reconstruction may be created. The reconstruction allows for the continued study of the site by archaeologists, preservation of the site, and access for those who are not qualified technical divers to explore it virtually.

Our original work in [20] proposed a multi-step process to identify and validate these sites. First, an AUV takes high-altitude, low-resolution scans of a large area. Second, the scans are fed into image processing software where potential sites are identified and ranked. Third, a path to visit the highest-ranked sites is planned and an AUV is deployed to take low-altitude, high-resolution images.

The goal of this work is to improve the image processing step in order to classify potential areas of interest more effectively. Specifically, this paper presents a number of contributions to the field of underwater robotics including:

- An archaeological site identification algorithm pipeline that processes AUV-obtained SSS data to yield locations of underwater archaeological sites of interest.
- Demonstrated increases in performance over standard CNN approaches due to the incorporation of conventional feature extraction methods.
- Experimental validation with a series of AUV deployments in Malta that produced actual site identifications.

The paper is organized as follows. Section 2 presents relevant background information on the use of AUVs in shipwreck search and mapping and the application of machine learning to this field. Section 3 gives an overview of the algorithm and an in-depth view of each stage of the pipeline. Section 4 details how the data was collected and how the algorithm performed. Finally, Sect. 5 discusses conclusions and future work.

2 Background

When surveying a large area to search for shipwrecks or other archaeological sites of interest, marine archaeologists often turn to SSS sensors [1]. SSS, an acoustic imaging technique, has a number of advantages over light-based approaches like video cameras. While SSS typically creates lower resolution imagery than cameras, acoustic waves do not attenuate as quickly in water and are able to capture large objects at a greater range in a single frame [23].

Once the sonar data has been collected, it can be converted to images which are analysed by archaeologists for potential sites of interest. In order to identify such regions, the archaeologists look for defined objects in the scans, using common attributes such as the size, shape, and texture of the object as well as the presence or absence of a shadow to differentiate archaeological artefacts from terrain features and noise in the data [20].

A number of different approaches exist for processing this image data automatically. First, conventional image processing techniques allow for the extraction of a wide variety of image features, such as contours, edges, and key points [14]. These features can be obtained efficiently through libraries like OpenCV [3]. Previous work has shown the effectiveness of these techniques on images extracted from SSS data. Chew, Tong, and Chia [5] use features like size and outline regularity to label objects as man-made or not, while Daniel et al. [6] use bounding boxes and object distinctiveness for classification.

However, in recent years, convolutional neural networks have emerged as the technique of choice for image processing tasks. State-of-the-art convolutional neural network models are trained on data sets like Microsoft COCO [12] and ImageNet [19] which contain hundreds of thousands or even millions of images. The vast quantities of data are necessary to properly fit the large number of parameters these models contain [9, 22]. If too little data is used, the model overfits, meaning that it is accurately able to classify the images used for training but not other examples of the same type.

Unfortunately, this volume of data is not always available, i.e. in marine archaeology where data is expensive and difficult to collect [8]. A number of different approaches exist for dealing with overfitting in a neural network, including batch normalization [10], transfer learning [16], dropout [24], and regularization [25].

Another technique, data augmentation, is often used to “generate” more data without having to collect more samples. Perez and Wang [15] propose a number of ways to do this, including traditional transformation techniques like shifting, rotating, flipping, shading, etc. and two neural network-based approaches, generative adversarial networks—which start with noise and produce images similar to the training data—and their own augmentation network—which takes two training images and combines them to produce a third.

A number of different machine learning techniques have been applied to the classification of SSS data. Previous work has demonstrated the effectiveness of machine learning for object detection in SSS images [4, 7, 17]. Neural network-based techniques have also been used with promising results [11, 13, 21].

Data augmentation has also been applied to SSS images. In order to augment their data and classify mine-like objects (MLOs), Barngrover et al. [2] drew polygons around each MLO and copied only the relevant pixels within that polygon onto examples of images without any features of interest. They also copied negative image pixels in the same shapes from one scan to another in order to account for any potential artefacts that may have been introduced in the augmentation process. This semi-synthetic data was then used with Haar-like and local binary pattern features as well as boosting to classify windows as containing MLOs or not.

In contrast to the previous works cited above, this work demonstrates how data augmentation and traditional image feature extraction methods can be combined and used to improve the performance of CNNs when applied to SSS data.

3 Intelligent Shipwreck Search

Searching for shipwrecks and other sites of archaeological interest in an unsurveyed area involves multiple rounds of AUV deployments combined with post-deployment image processing and AUV path planning. First, an area is chosen to be surveyed based on historical data. High-altitude AUV lawnmower paths are planned to cover the entire area to obtain low-frequency, low-resolution SSS. The sonar scans are then fed into a sonar mapping software that converts the raw data into images to be analysed.

The images are then examined for potential areas of archaeological interest. Once such regions have been identified, new low-altitude AUV paths are planned to obtain high-frequency, high-resolution sonar and video data. This new data can be used to confirm the type of site (e.g. plane wreck), generate maps for additional AUV path planning, and create three-dimensional wreck models using photogrammetry.

Traditionally, in the site identification stage, an archaeologist manually looks through each sonar image, a process which can be time consuming for large data sets and is largely dependent on the experience and skill of the archaeologist. This paper demonstrates that the same process can be done automatically using a site identification algorithm pipeline that takes as input SSS images and outputs potential sites of interest with their corresponding locations in the images. The next section provides details of this pipeline.

3.1 Site Identification Algorithm Pipeline

The pipeline seen in Fig. 2 effectively identifies archaeological sites of interest in a SSS image. First, a scan I_{scan} is discretized into N 128×128 pixel sub-images, the set of which is termed I_{chop}. Each of the N sub-images i_{chop} is considered to be an

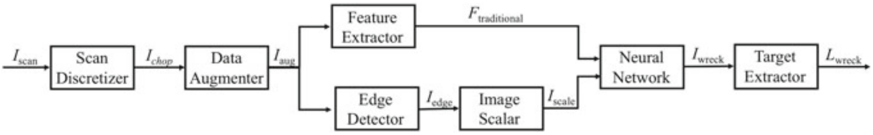


Fig. 2 Block diagram of the sonar scan processing pipeline

area within which a potential site could be located. As such, each sub-image i_{chop} is passed through a data augmenter that generates S moderately transformed images of size 200 × 200 pixels.

Each augmented sub-image, the set of which is termed I_{aug} with cardinality $|I_{aug}| = NS$, is inserted into two different processes, one that extracts image features $F_{traditional}$ with traditional image processing techniques and one that filters the image to detect edges and scales it back to a 128 × 128 pixel image, the set of which is called I_{scale}. Both $F_{traditional}$ and I_{scale} are passed to a neural network for processing. The neural network labels each of the NS sub-images in I_{scale} as either containing a site of interest or not, and outputs the set of images including such a region I_{wreck}.

Finally, the target extractor function outputs L_{wreck}, which contains the locations of only those sub-images i_{chop} in I_{chop} identified by the target extractor as containing a site of interest. Details of each step in the pipeline are provided below.

3.2 Scan Discretizer

When given a new SSS image, I_{scan}, the algorithm first uniformly and evenly divides the scan into 128 × 128 pixel sub-images. The size 128 × 128 pixels was chosen because it is small enough to allow for smaller sites of interest to take up a significant per cent of the image, while large enough to allow for most of a larger site to be captured. If the site is too large, data augmentation described in Sect. 3.3 is sufficient to allow all regions of a partially cut-off area of interest to be identified (see Fig. 5b).

Associated with each sub-image i_{chop} is a 4-tuple label that includes the X and Y coordinates of the top-left corner as well as the width and height of the area. This label will be outputted at the end of the pipeline and can be used to help geo-localize the sub-image if it is identified as containing a site of interest.

3.3 Data Augmenter

Data augmentation serves a number of purposes within the proposed algorithm. First, it allows for the creation of a larger data set for training purposes, the details of which are described in Sect. 4.3. Second, augmenting the data inputs to the trained network (e.g. at testing time) increases the likelihood that it will be correctly classified by the pipeline.

Each sub-image i_{chop} is passed through data augmentation S times to create S augmented versions of i_{chop}. Transformations are used to create the augmented versions of the sub-images, specifically scaling, translating, flipping, and shearing. Details of these transformations are provided below.

The first transformations are **scaling** and **translating** which are applied in conjunction with each other. First, a value for the scaling factor γ is selected by randomly sampling a log-uniform value between $\frac{1}{2}$ and 2. Second, the values for the width and height of a sub-image are calculated such that when later scaled by γ, they will yield a size of 200×200. Next, the coordinates of the top-left corner of the sub-image to be extracted from the full scan are randomly selected such that the resulting box defined by the top-left corner, width, and height will encapsulate the entire region i_{chop}. Finally, the sub-image is extracted from the full scan and scaled by γ to a size of 200×200 pixels.

Two types of **flipping** may be applied to each sub-image, a horizontal flip and a vertical flip. This is highly relevant for SSS images because features protruding from the seafloor block the sonar waves, creating an "acoustic shadow" that extends away from the nadir, the area directly below the sensor. Therefore, features on the left side of a scan image extend shadows to the left, and features on the right side of the image extend shadows to the right, (see Fig. 5a, b). In order to remove any correlation between the side of the scan and the label, all regions on the right side of the nadir of the scan are flipped horizontally. Additionally, to further augment the data, each image has a 50% chance of being flipped vertically.

Finally, in order to further distort the sub-image, each one is either horizontally or vertically **sheared**. The result of OpenCV's warpAffine, which is used for this task, is a parallelogram. In order to maintain consistent sub-image dimensions (200×200), the triangles on each side of the parallelogram are ignored and only the middle 200×200 square is passed to the next stage.

3.4 Image Feature Extractor

Seven additional features are extracted from each sub-image in I_{aug} by using standard vision processing techniques in order to augment the information extracted by the neural network. The seven features included are the number of corners, number of lines, number of blobs, number of ORBs, number of contours, minimum pixel intensity value, and maximum pixel intensity value. Most of the features are extracted using functions provided in the OpenCV Python library.

Corners are found using Harris corner detection which uses a windowing function multiplied with the x and y gradients of the image to determine which pixel regions contain corners. Lines are detected using Canny edge detection, an algorithm that takes the gradient of an image, thresholds on intensity, and classifies the strength of the edges. Contours are found by finding connected boundaries with the same pixel colour or intensity. Blob detection operates by determining contours and filtering them by certain features, like circularity and convexity [14]. Oriented FAST and Rotated BRIEF (ORB) is a feature detector developed by OpenCV that picks out important features for use in applications like panorama stitching [18].

Each of these features is extracted with a specific goal in mind. Corners and lines highlight wrecks, since man-made objects are almost always made with defined lines and corners, but natural features having gone through much more erosion and weathering, have smoothed out their sharp edges. Blobs help identify rocks, since, as mentioned above, natural objects are often eroded and formed into rounded objects. ORBs and contours measure the number of important features in each image. Because archaeological sites of interest stand out from the background, they are most often detected in positive images. The minimum and maximum pixel intensities highlight the presence of a large object that casts a large shadow.

Due to high levels of noise in the images generated from the sonar data, all sub-images were first blurred with Gaussian and median blurring before attempting to detect the seven features. This makes it difficult to detect smaller objects, but vastly reduced the number of false positive identifications by the feature extraction.

The output of the image feature extractor is the feature set $F_{traditional}$, which includes values for the seven features associated with each of the NS sub-images.

3.5 Edge Detector

Using edge detection on the sonar scan sub-images dramatically improves site identification performance. In this stage of the pipeline, each of the NS sub-images in I_{aug} is passed through the median and Gaussian blurring filters, before being passed through a Canny edge detection algorithm provided in OpenCV. The output of this stage is a new set of sub-images I_{edge}.

3.6 Image Scalar

The sub-images in I_{edge} are of dimension 200 × 200 pixels. This size was found to be effective for feature extraction, but too large for a practical neural network training duration. To reduce training time without loss of neural network performance in terms of precision and recall, each sub-image is resized to be 128 × 128 pixels and outputted in the sub-image set I_{scale}.

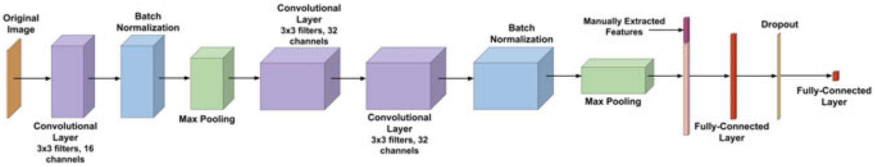

Original Image
Convolutional Layer 3x3 filters, 16 channels
Batch Normalization
Max Pooling
Convolutional Layer 3x3 filters, 32 channels
Convolutional Layer 3x3 filters, 32 channels
Batch Normalization
Max Pooling
Manually Extracted Features
Fully-Connected Layer
Dropout
Fully-Connected Layer

Fig. 3 Neural network architecture used in the pipeline

3.7 Neural Network

Each sub-image in I_{scale} and the corresponding feature set $F_{traditional}$ are then used as an input to the neural network (NN). The architecture for the network is shown in Fig. 3, and is based on an architecture proposed by Perez and Wang [15]. Though several modifications to this architecture were explored and implemented, no difference in performance was observed.

The first series of steps in the NN architecture form a convolutional neural network (CNN):

1. Convolutional layer with 3 × 3 filters and 16 channels. ReLU activation.
2. Batch normalization.
3. Max pooling with 2 × 2 filters and 2 × 2 stride.
4. Convolutional layer with 3 × 3 filters and 32 channels. ReLU activation.
5. Convolutional layer with 3 × 3 filters and 32 channels. ReLU activation.
6. Batch normalization.
7. Max pooling with 2 × 2 filters and 2 × 2 stride.

In order to combat overfitting in the network, regularization is also applied to each of the convolutional layers. The output of the final max pooling layer is then flattened in preparation for the next stage of the pipeline.

The flattened output of the convolutional neural network is then concatenated with the output of the image feature extractor and fed into the following architecture:

1. Fully connected layer with output dimension 1024. ReLU activation.
2. Dropout.
3. Fully connected layer with output dimension 2.

The final output of the last step of the NN is a label for each of the NS sub-images in I_{scale}, i.e. whether the sub-image contains a site or not. Hence the NN function block outputs I_{wreck}, the subset of the NS images that were labelled as containing a site of interest.

3.8 Scan Target Extractor

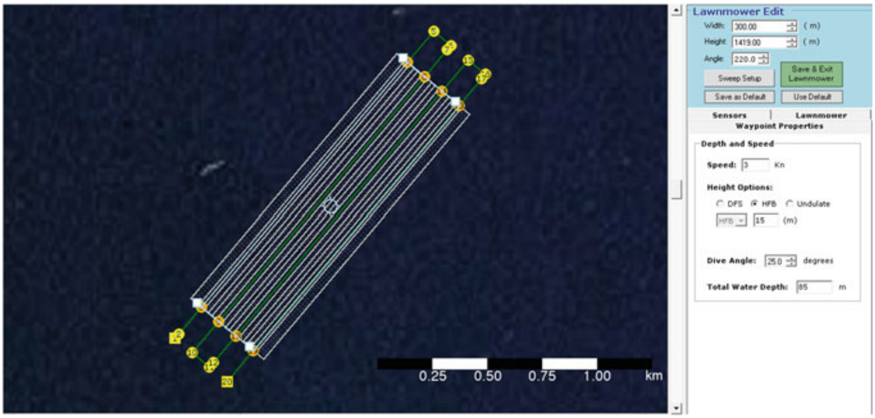
The target extractor function outputs L_{wreck}, which contains the locations of sub-images i_{chop} in I_{chop} that contain a site of interest. A sub-image i_{chop}'s location is included in L_{wreck} if some minimum fraction of the S sub-images generated by augmenting i_{chop} are labelled by the neural network as containing a site of interest. In general, the threshold fraction is $\frac{S}{2}$.

4 Experiments

Several AUV deployments were conducted during actual archaeological site search expeditions in the Mediterranean Sea in which SSS data was collected and used to validate the performance of the site identification algorithm pipeline.

4.1 Data Collection

Deployments were conducted with a BF12-1006 Bluefin AUV equipped with an Edgetech 2205 Dual Frequency 600/1600 kHz SSS module. High-altitude lawnmower pattern paths, e.g. shown in Fig. 4, were executed for each deployment. Two specific data sets were collected, using the same AUV but taken a year apart, (i.e. 2015 and 2016), and labelled throughout the rest of the paper as data set A and data set B. The survey areas for these data sets were 4 km by 4 km. Example annotated images of the SSS data collected are shown in Fig. 5a, b.



Width: 300.00 (m)
Height: 1419.00 (m)
Angle: 220.0

Speed: 3 Kn

Height Options:
○ DFS ● HFB ○ Undulate
HFB 15 (m)

Dive Angle: 25.0 degrees

Total Water Depth: 85 m

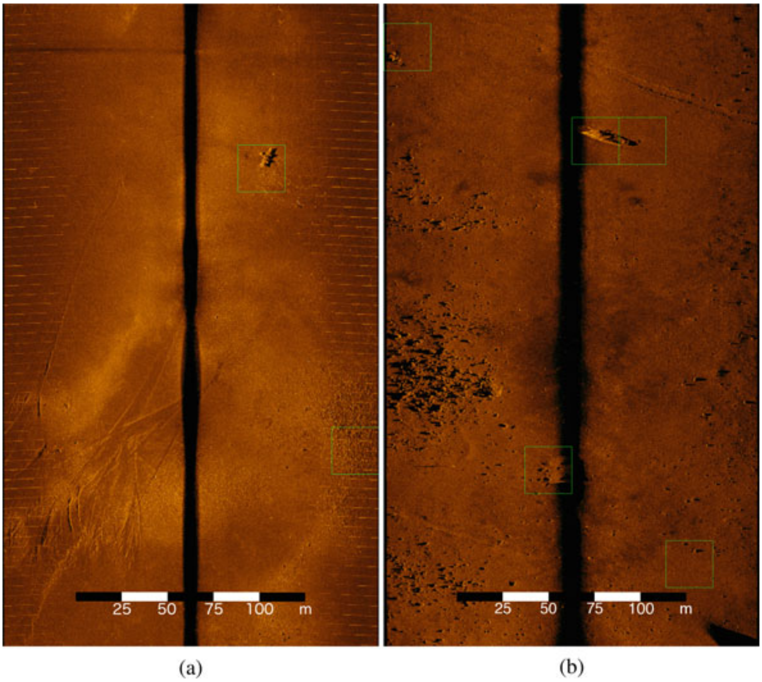
Fig. 4 Screenshot from VectorMap Software

4.2 Labelled Data Set Creation

The data used in the following experiments was labelled by both an experienced archaeologist and student researchers. Labelling areas of archaeological interest is extremely difficult, even for experienced humans. For example, a recent data set collected by the team in 2017 was labelled by two different sets of researchers. Only 26% of the proposed sites were proposed by both groups.

This discrepancy is due to a number of difficulties associated with identifying regions of interest. First, the resolution of the scans is about 19.5 cm^2/pixel, making fine details challenging to make out. In addition, noise generated in the process of taking sonar data and converting it to an image can create artefacts that look like sites of interest or obscure the underlying data. Finally, debris, rocks, and other features of the natural terrain contain many of the same features as objects of interest.

Unfortunately, these features are not uniform across data sets and can depend on a number of factors, including the sonar sensors used, the area being surveyed, and the ocean conditions when the data was taken. Notice that Fig. 5a, from data set A,



(a) (b)

Fig. 5 Examples of scans taken from the **(a)** A and **(b)** B data sets labelled with predictions from the site identification algorithm. Note that each image corresponds to only a fraction of the length of one path line from a lawnmower pattern such as the one shown in Fig. 4. Image courtesy of Vulcan Inc./University of Malta

contains regular linear artefacts of the sonar scanning process, while Fig. 5b, from data set B, features a rocky terrain.

4.3 Neural Network Training

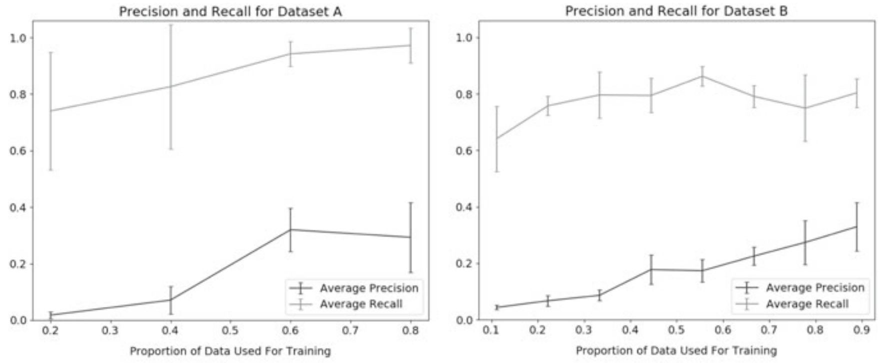
A number of strategies were employed to effectively train the neural networks. First, regions within the scans were selected for training. These included all areas labelled as containing a site of interest as well as a number of hand-picked areas with no sites. These negative regions were selected based on whether or not they contained a feature of the terrain that had not yet been included, as well as whether or not similar regions had previously been consistently misclassified.

Since the results of one trial were used to inform the data used for the next, the images used for validation were randomly selected each time. This minimized bias in the training data by forcing the labellers to consider what data needed to be included to represent as many regions of as many scans as possible.

However, the total number of images in the data set was far lower than is necessary for training the above-described CNN. With the goal of creating a larger data set, each image was passed through the data augmenter. Because of the lack of examples of archaeological sites of interest, the training data set generation process led to the selection of far more regions without sites of interest. To address this, the number of augmented copies of each negative image was reduced so that the total number of positive and negative examples fed to the neural network was balanced. Augmenting both the images containing objects of archaeological value and those that did not have the added effect of reducing any potential bias correlated to features of the augmented image that may have been introduced during the augmentation process. For data set A, this resulted in augmented 1124 images used for training and validation, while for data set B, augmented 6024 images were used.

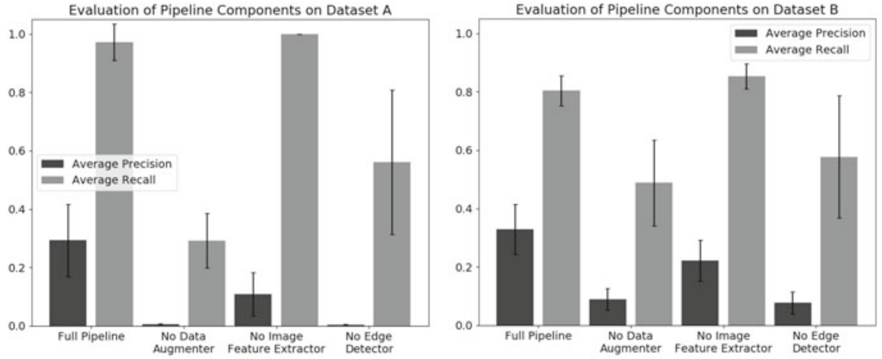
4.4 Results

This section presents the site identification algorithm pipeline performance evaluation experiments. While pipeline performance was measured using precision and recall, two sets of results are presented that ground these measurements in our problem domain. For each trial, a proportion of the sonar scans were selected for training versus validation. Once the network was trained, each validation image was put through the pipeline, and the number of true positives, false positives, and false negatives was counted. Because the scans used for validation were selected randomly and the sites of interest are not evenly distributed across the scans, these three numbers were aggregated across five different rounds to compute precision and recall for a single trial of the algorithm. Six trials of each configuration of the algorithm were averaged to compute the results seen in Figs. 6 and 7.



(a) (b)

Fig. 6 Validation precision and recall as a function of the proportion of **(a)** data set A or **(b)** data set B used for training



(a) (b)

Fig. 7 Validation precision and recall for the algorithm with different blocks of the pipeline removed for **(a)** data set A and **(b)** data set B

First, precision and recall were measured for cases of increasing fractions of a data set being used for training versus testing. This provides a measure of one's ability to start an AUV search expedition in a new area, and allow training to be done on some fraction of initial images before being confident that the pipeline will be successful in identifying sites on the remaining images. Figure 6 illustrates pipeline performance for increasing fractions of training data. As shown in the figures, for data set A, the precision reached 29.34% and the recall reached 97.22%, while for data set B, the precision reached 32.95% and the recall reached 80.39%.

Second, precision and recall were measured for cases with and without some stages of the pipeline to highlight the individual component effects. These components—the Data Augmenter, the Image Feature Extractor, and the Edge Detector—were removed from the pipeline. The network was retrained on 80% and 88.89% of data sets A and B, respectively. The bar graphs in Fig. 7 show the precision and recall of these different configurations. The above figures demonstrate how crucial all three of these stages are in improving the algorithm performance.

Precision and recall are both important values to consider when evaluating the effectiveness of the algorithm. A high recall means that an archaeologist is less likely to miss a potential region of interest, while a high precision reduces the number of sites an archaeologist must revisit. An algorithm that performs well when assessed by both metrics allows archaeologists to explore more sites that could lead to discoveries while spending less time at sites that will not. Though our algorithm exhibits a low precision, it filters out 99.79% of the area in data set A and 99.31% of the area in data set B, highlighting the efficiency of using such an algorithm.

5 Conclusions and Future Work

This work presents a pipeline that identifies archaeological sites of interest in SSS data obtained with an AUV. The proposed algorithm presents data augmentation and image feature extraction as two approaches to increase performance over standard CNN approaches. In addition, the paper provides experimental validation in the form of AUV deployments off the coast of Malta that led to the discovery of sites of archaeological significance.

The algorithm presented achieves a precision and recall of 29.34% and 97.22% on one data set and 32.95% and 80.39% on the second. These results demonstrate that the proposed pipeline identifies a majority of the archaeological sites tested, but misclassifies some noise and features of the natural terrain as regions of interest. Due to the low precision, this algorithm should be used to inform archaeologists about areas to consider revisiting by giving them a smaller sample size to review.

This paper discusses an automated pipeline for identifying sites of archaeological interest from SSS images. Future work may include testing the algorithm on a wider variety of data, including scans from different altitudes and different angles. Because this data is expensive to collect, another area for future work is simulating the creation of these scans in a software. In addition, the pipeline presented can be integrated into the AUV intelligent search system.

Acknowledgements We would like to thank Jeffrey Rutledge, Samantha Ting, Russell Bingham, Andrew Pham, Kolton Yager, Bonita Galvan, and Mitchell Keller for helping us deploy the robot. This work was performed in part at Claremont Colleges Robert J. Bernard Biological Field Station. This material is based upon work supported by National Science Foundation under Grant No. 1460153.

References

1. Atallah, L., Shang, C., Bates, R.: Object detection at different resolution in archaeological side-scan sonar images. Eur. Ocean **2005** (2005). https://doi.org/10.1109/OCEANSE.2005.1511727
2. Barngrover, C., Kastner, R., Belongie, S.: Semisynthetic versus real-world sonar training data for the classification of mine-like objects. J. Ocean Eng. **40**, 48–56 (2015)
3. Bradski, G., Kaehler, A.: Learning Opencv, 1st edn. O'Reilly, Sabostopol, CA (2008)
4. Chang, R., Wang, Y., Hou, J. et al.: Underwater object detection with efficient shadow-removal for side scan sonar images. OCEANS 2016—Shanghai (2016). https://doi.org/10.1109/OCEANSAP.2016.7485696
5. Chew, A., Tong, P., Chia, C.: Automatic detection and classification of man-made targets in side scan sonar images. Symposium on Underwater Tech (2007). https://doi.org/10.1109/UT.2007.370841
6. Daniel, S., Le Léannec, F., Roux, C., et al.: Side-scan sonar image matching. J. Ocean Eng. **23**, 245–259 (1998)
7. Dura, E., Zhang, Y., Liao, X., et al.: Active learning for detection of mine-like objects in side-scan sonar imagery. J. Ocean Eng. **30**, 360–371 (2005)
8. Gambin, T.: Side scan sonar and the management of underwater cultural heritage. In: Formosa, S. (ed.) Future Preparedness: Thematic and Spatial Issues for the Environment and Sustainability. Msida, Malta (2014)
9. He, K., Zhang, X., Ren, S. et al.: Deep residual learning for image recognition (2016). arXiv:1512.03385
10. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv:1502.03167
11. Langner, F., Knauer, C., Jans, W. et al.: Side scan sonar image resolution and automatic object detection, classification and identification. OCEANS 2009-EUROPE (2009). https://doi.org/10.1109/OCEANSE.2009.5278183
12. Lin, T.Y., Maire, M., Belongie, S. et al.: Microsoft COCO: common objects in context (2014). arXiv:1405.0312
13. McKay, J., Gerg, I., Monga, V. et al.: What's mine is yours: pretrained CNNs for limited training sonar ATR (2017). arXiv:1706.09858
14. Nixon, M., Aguado, A.: Feature Extraction and Image Processing for Computer Vision, 3rd edn. Academic Press (2012)
15. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning (2017). arXiv:1712.04621
16. Razavian, A.S., Azizpour, H., Sullivan, J. et al.: CNN features off-the-shelf: an astounding baseline for recognition (2014). arXiv:1403.6382
17. Reed, S., Petillot, Y., Bell, J.: An automatic approach to the detection and extraction of mine features in Sidescan sonar. J. Ocean Eng, **28**, 95–105 (2003)
18. Rublee, E., Rabaud, V., Konolige, K., et al.: ORB: an efficient alternative to SIFT or SURF. Int. Conf. Comput. Vis. (2011). 10.1.1.370.4395
19. Russakovsky, O., Deng, J., Su, H. et al.: ImageNet large scale visual recognition challenge (2015). arXiv:1409.0575
20. Rutledge, J., Yuan, W., Wu, J., et al.: Intelligent shipwreck search using autonomous underwater vehicles. Int. Conf. Robot Autom. (2018). https://doi.org/10.1109/ICRA.2018.8460548
21. Shang, C., Brown, K.: Feature-based texture classification of side-scan sonar images using a neural network approach. Electron. Lett. **28**, 2165–2167 (1992)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015). arXiv:1409.1556

23. Singh, H., Adams, J., Mindell, D., et al.: Imaging underwater for archaeology. J. Field Archaeol. **27**, 319–328 (2000)
24. Srivastava, N., Hinton, G., Krizhevsky, A., et al.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
25. Wang, B., Klabjan, D.: Regularization for unsupervised deep neural nets (2017). arXiv:1608.04426

Ridge-Tracking for Strawberry Harvesting Support Robot According to Farmer's Behavior

Ayanori Yorozu, Genya Ishigami, and Masaki Takahashi

1 Introduction

The increase in global population has led to an increase in food demand, shortage of agricultural workforce due to aging, and a decline in the quality of harvested products. To overcome these issues, some researches and developments of agricultural robots have been actively conducted to replace the workforce [1, 2]. In addition, for increasing the crop yield and improving the quality of crops, development of smart farming and automated agricultural technology is underway. These methods and technologies aim at constructing database of agricultural activities such as watering, weeding, harvesting, and crop yield, and automating production management [3, 4].

In recent years, a highly accurate outdoor localization for robots has been made possible by the real-time kinematic global navigation satellite system (RTK-GNSS). Techniques using the RTK-GNSS to autonomously move within the field while following a pre-planned path from the field map have been proposed [5]. Especially in large-scale farms, large-sized tractor automation technology and autonomous harvesting robots that move in the field by localization, observe the degree of crop maturity, and harvest the crops using robot arms have been developed. On the other hand, approximately 70% of agricultural producers in Japan possess dispersed farms

---

A. Yorozu (✉)
Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan
e-mail: ayanoriyorozu@keio.jp

G. Ishigami
Faculty of Science and Technology, Department of Mechanical Engineering, Keio University, Yokohama, Japan
e-mail: ishigami@mech.keio.ac.jp

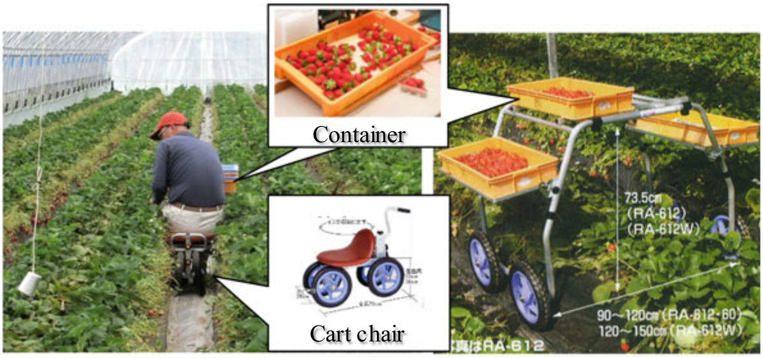
M. Takahashi
Faculty of Science and Technology, Department of System Design Engineering, Keio University, Yokohama, Japan
e-mail: takahashi@sd.keio.ac.jp

© Springer Nature Singapore Pte Ltd. 2021

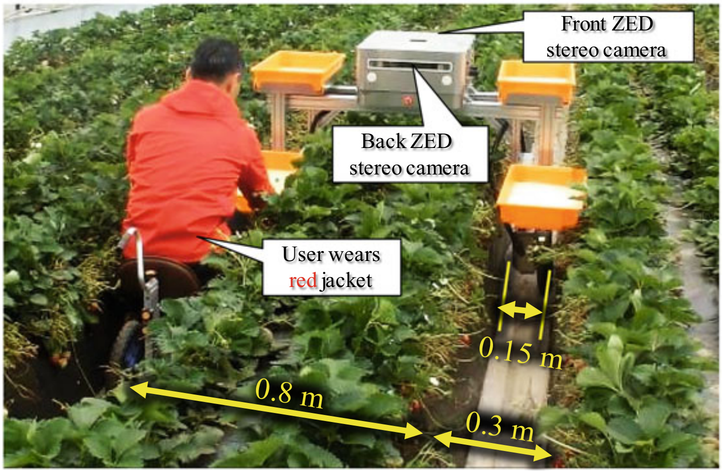
G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_17

on a medium and small scale (less than 5 ha), and they often cultivate a variety of crops on these lands. Due to the scale, it is difficult to introduce large-sized agricultural machines to these farms [6]. In these medium/small-scale farms, farm works such as watering, weeding, and harvesting are performed by human workers. For example, as shown in Fig. 1a, during harvesting, the farmer moves through the field carrying a container or a cart for loading the harvested crops, identifies the mature crops, and loads them in the container. Thus, a lot of labor is required to transport containers loaded with crops and push the cart or containers. Therefore, production management and agricultural support technology customized for small/medium-scale decentralized agriculture, different from the large-scale types, are required.

In this study, we focus on small/medium-scale farms of 2–3 ha, which account for the aforementioned 70% of Japan's farms. A consortium led by Japan Research


Container
Cart chair
73.5cm
(RA-612)
(RA-612W)
90～120cm (RA-612-60)
120～150cm (RA-612W)
RA-612

(a) Before: Strawberry harvesting by farmers


Front ZED
stereo camera
Back ZED
stereo camera
User wears
red jacket
0.15 m
0.8 m
0.3 m

(b) After: MY DONKEY® harvesting support

Fig. 1 Strawberry harvesting support

Institute, Ltd., proposes a compact multi-functional agricultural robot called "MY DONKEY$^{®}$" [7]. As shown in Fig. 1b, this robot supports transportation of harvested crops and records the work on the farm done by the user and the crop yield of the land on the field map, based on localization, while maneuvering closer to the user. As opposed to several of the previous works on automation technology of the agricultural robot, this study focuses on designing a user's behavior-based harvest support system in a furrowed field.

To avoid collision with ridges and crops in the furrowed field, the robot must be able to detect the navigable ridges and furrows using onboard sensors and follow the ridge by the means of a robot coordinate system. In this study, a center for ridge detection using point cloud data from a stereo camera mounted in the robot and a safe ridge-tracking control to avoid ridges and crops are proposed. Furthermore, to realize smooth harvesting support, a ridge-tracking control according to farmer's (user's) behavior, such as moving, harvesting, and loading the harvested crops in the robot, is proposed. For example, the robot moves to maintain a certain distance from the user, when the user moves and harvests the crops; and then stops and waits until it has been completely loaded with the harvested crops upon recognizing the farming work contents of the user. In this study, we propose a ridge-tracking control framework based on the fuzzy set theory, which evaluates and integrates multiple situations and functions. The proposed system is applied to strawberry harvesting, and experiments using "MY DONKEY$^{®}$" on a strawberry farm are carried out.

2 Strawberry Harvesting Support Using "MY DONKEY$^{®}$"

2.1 Strawberry Harvesting

Figure 1a shows a typical strawberry harvesting activity. In the strawberry harvesting work, the farmer uses a cart of multiple harvesting containers. The farmer sits on a cart chair, picks the strawberries, moves while loading them into the container, and again stands up to arrange the containers filled with harvested strawberries. During the harvesting work, the farmer moves through the field carrying a container or cart for loading the harvested crops. It is desirable for the robot to move through the furrowed field and maintain a certain distance from the user when the latter is sitting, and to stop and wait when they are standing to load the containers on to the robot.

Therefore, in this study, the user's behavior is identified from the position of user (standing or sitting), and the ridge-tracking controller is designed based on the user's behavior.

2.2 System Configuration

Figure 1b shows a robot system configuration for the strawberry harvesting support. The robot operating system (ROS) framework is used for the robot system. As shown in Fig. 1b, stereo cameras (ZED©Stereolabs), which can be used outdoors and can obtain 3D point cloud data over a wide range, are equipped on the front and backside of the base module of the robot for ridge and user detection. In this study, for the robot to detect the user to follow, the user wears a red jacket.

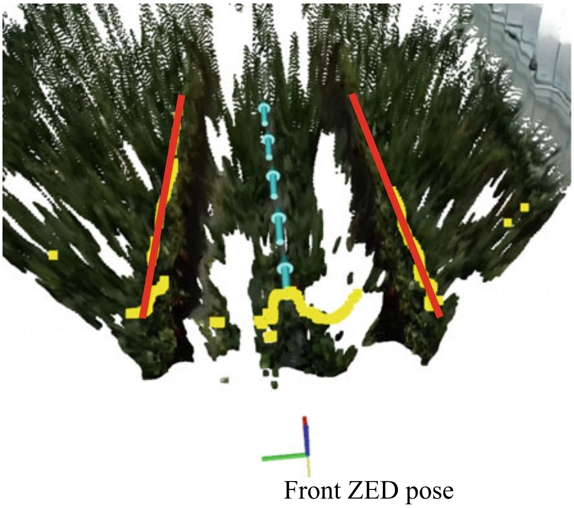
The proposed system determines the velocity command to track the ridge according to the user's behavior based on obtained sensor data. The robot has a four-wheel, independent drive system. It is possible to steer each wheel separately. Additionally, the mass meters were mounted to measure the yield of strawberries.

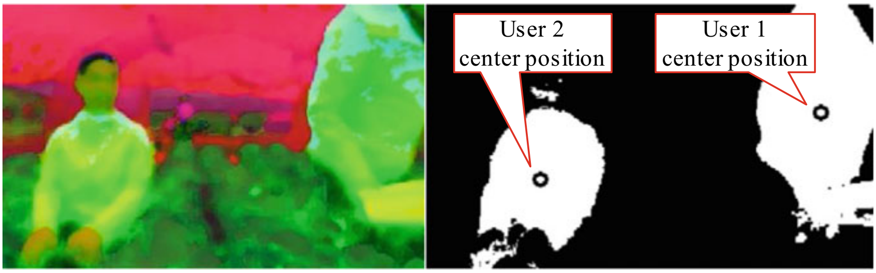
3 Ridge-Tracking Control According to Farmer's Behavior

3.1 Ridge Detection

Figure 2 shows the 3D point cloud data (PCD) extracted from the ZED camera in the strawberry farm. As shown in Fig. 2, it can be inferred that it is difficult to acquire the shape of the ridge accurately because of the noise and data loss in PCD. On the other hand, the contours of both the adjacent ridges can be acquired at relatively low data loss. From experimental data, the center line of the ridge for the robot to follow is detected from the lines at the edge of both the adjacent ridges using the Hough

Fig. 2 Ridge detection using front ZED camera





(a) HSV image (b) User detection

Fig. 3 User detection using back ZED camera

transform in the robot coordinate system, because the ridges of the strawberry rows are straight and furrowed at regular intervals.

First of all, as represented by yellow points in Fig. 2, 2D point data at the certain height are extracted from 3D PCD. Then, as represented by the straight red lines in Fig. 2, ridge lines are detected from the 2D point data using Hough transform. Finally, as represented by light blue arrows in Fig. 2, the center of line of the ridge, along which the robot follows, is calculated from both the detected left- and right-side lines.

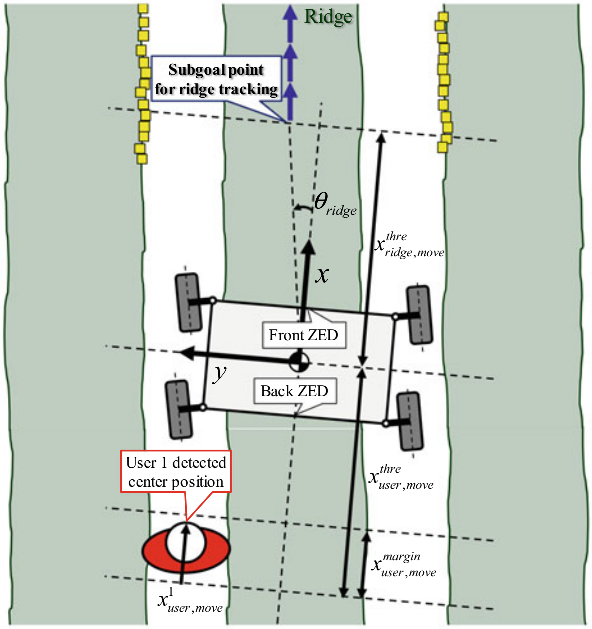
3.2 User Detection

The position of the users who wear red jackets is calculated using the back ZED camera. At first, as shown in Fig. 3a, an RGB image from ZED camera is transformed into an HSV image, so that it could be relatively easily extracted from the red jacket. Then, as shown in Fig. 3b, the hue is adjusted to extract only the area of red jackets of the users. In addition, as represented by "o" in Fig. 3b, the jacket color area is clustered and the image centers of gravity of the color region clusters whose pixel area is larger than the threshold value are calculated. Finally, the center position of users x^k_{user}, y^k_{user}, z^k_{user} in the robot coordinate system are calculated from the depth image of the ZED camera's stereo vision corresponding to the detected image center of gravity. The superscript $k = 1$ or 2 indicates the number of detected users.

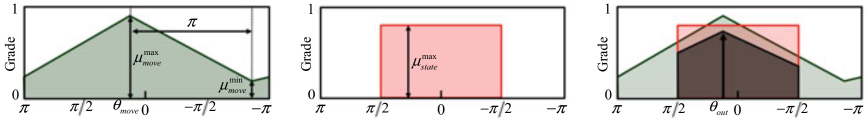
3.3 Ridge-Tracking Control Using Fuzzy Potential Method

To achieve smooth strawberry harvesting support, we propose ridge-tracking control that functions by the user's behavior based on the detection of users, as shown in Sect. 3.2. When the robot and user are moving between the furrowed field, it is possible to identify the user's behavior—moving and harvesting or loading containers—based on the state of the user (standing or sitting), which is nothing but the height of the center of gravity of user z^k_{user}. In this study, we propose ridge-tracking control using the fuzzy set theory, to evaluate and integrate multiple conditions simultaneously, based on the height of the center of gravity of user z^k_{user} and the environment (ridges) surrounding the robot. For this study, we use the fuzzy potential method (FPM) [8], in which the membership functions (MFs) for each target action or element are generated. The horizontal axis of the MF is the yaw angle of the robot, and the vertical axis of MF is the grade that indicates the priority of the target action with respect to the angle. Then, the motion (velocity command) is decided by integrating MFs.

Figure 4 shows an example of the determination of velocity command (v_x, v_y, ω_z) by the proposed method for strawberry harvesting. First, the MF, μ_{move}, for moving to track the ridge and maintain a fixed distance from the user is generated. Next, the MF, μ_{state}, for chaining the motion according to the user's behavior is generated.


Ridge
Subgoal point for ridge tracking
θ_{ridge}
$x^{thre}_{ridge,move}$
x
Front ZED
y
Back ZED
User 1 detected center position
$x^{thre}_{user,move}$
$x^{margin}_{user,move}$
$x^{1}_{user,move}$

(a) Top view of strawberry harvesting scene



(b) MF μ_{move} (c) MF μ_{state} (d) Mixed MF μ_{mix}

Fig. 4 Ridge-tracking control according to farmer's behavior using fuzzy potential method

Then, the velocity command is calculated based on the MF, μ_{mix}, and integrating μ_{move} and μ_{state}. Finally, the wheel speed and angle control of each wheel unit is performed based on the velocity command.

3.3.1 MF to Move Toward Center of Ridge

The MF μ_{move} for moving to track the detected ridge and maintain a fixed distance from the detected user position in the robot coordinate system is generated.

As shown in Fig. 4a, the intersection point of the detected ridge center line and equation $x = x^{thre}_{ridge,move}$ is set as a subgoal, which the robot follows in the robot coordinate system. The relative angle of the subgoal from the front of the robot is denoted by θ_{ridge}. The relative distance to maintain from the user is $x^{thre}_{user,move}$. The MF, μ_{move}, is generated as a triangular MF, so that the robot moves forward in the θ_{move} direction as the user approaches the robot for moving and harvesting. Figure 4b shows an example of the MF, μ_{move}. The angle of vertex, θ_{move}, of the MF, μ_{move}, is determined to move toward the subgoal without collision, based on the detected ridge.

3.3.2 MF to Consider State

An MF μ_{state}, for chaining the motion according to the user's farming task is generated based on the user's detected height of center of gravity z^k_{user}.

As shown in Fig. 4c, a convex MF with grade $\mu^{\max}_{state}$ is an angular range from $-\pi/2$ to $\pi/2$, because the users are harvesting continuously as they move forward in the strawberry rows. The grade $\mu^{\max}_{state}$ is designed to slow down and be stationary for loading yield crops or to move and maintain a certain distance from the user according to the height of the user z^k_{user}.

3.3.3 Velocity Command Based on Mixed MF

As shown in Fig. 4d, the velocity command (v_x, v_y, ω_z) is calculated based on the MF, μ_{mix}, and integration of μ_{move} and μ_{state}.

The moving direction, θ_{out}, and the translational moving speed, v_{out}, are calculated as follows based on the mixed MF.

$$\theta_{out} = \arg\max \mu_{mix}(\theta) \tag{1}$$

$$v_{out} = v^{\max}_{robot} \mu_{\max}(\theta_{out}) \tag{2}$$

where $v^{\max}_{robot}$ is the maximum translational speed of the robot. Then, the velocity command is calculated as follows.

$$v_x = v_{out} \cos\theta_{out} \tag{3}$$

$$v_y = v_{out} \sin\theta_{out} \tag{4}$$

$$\omega_z = \begin{cases} K_\omega \theta_{out} & if \ \ |K_\omega \theta_{out}| < \omega_{robot}^{\max} \\ \omega_{robot}^{\max} & else\ if\ K_\omega \theta_{out} > \omega_{robot}^{\max} \\ -\omega_{robot}^{\max} & else \end{cases} \tag{5}$$

where $\omega_{robot}^{\max}$ is the maximum angular speed and K_ω is an angular velocity gain.

4 Experiments

We carried out the experiments using MY DONKEY® robot in a strawberry farm. As shown in Fig. 5, we verified the strawberry harvesting support of two users. Table 1 shows the system parameters.

In this study, we created a two-dimensional occupancy map in advance on the ROS gmapping package using 2D point cloud data transformed from 3D point cloud data of a front ZED camera. The localization of the robot was performed using the adaptive Monte-Carlo localization in the field coordinate system during the movement for harvesting support.

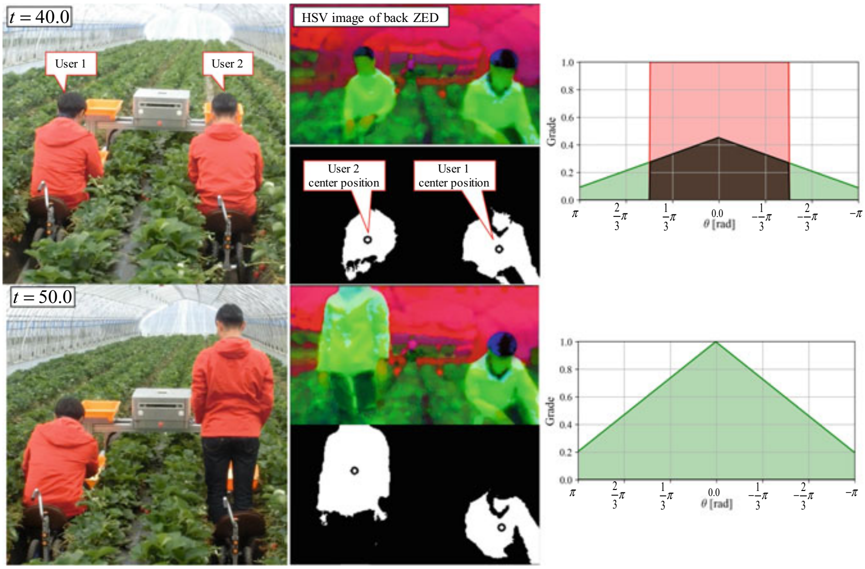
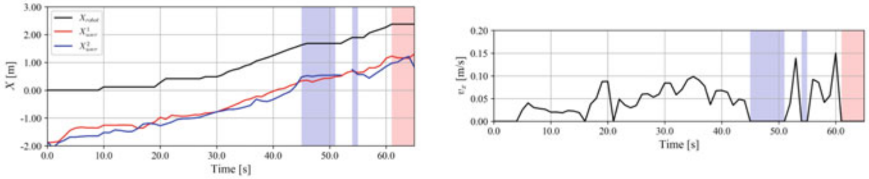

$t = 40.0$
User 1
User 2
HSV image of back ZED
User 2 center position
User 1 center position
$t = 50.0$
Grade
1.0
0.8
0.6
0.4
0.2
0.0
π
$\frac{2}{3}\pi$
$\frac{1}{3}\pi$
0.0
θ [rad]
$-\frac{1}{3}\pi$
$-\frac{2}{3}\pi$
$-\pi$

Fig. 5 Actual situations, user detection results and MFs in strawberry harvesting with two users

Table 1 System parameters

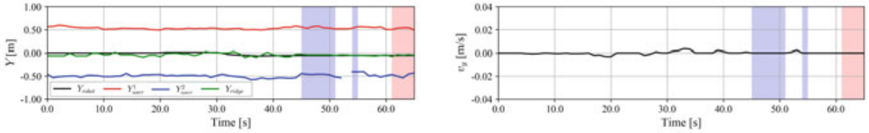
Parameters	Values
Subgoal distance $x_{ridge,move}^{thre}$	1.5 m
Maintain distance from user $x_{user,move}^{thre}$	−1.4 m
Maximum translational speed $v_{robot}^{\max}$	0.25 m/s
Maximum angular speed $\omega_{robot}^{\max}$	0.05 m/s
Angular velocity gain K_{ω}	0.5
Wheel radius r_{wheel}	0.135 m
Sampling time Δt	0.05 s

Figure 5 shows examples of the user detection results and MFs. Figure 6a–c shows the time history of robot localization, ridge-detected position (subgoal position), and user-detected positions in the field coordinate system. In Fig. 6a, b, the black line indicates the robot position, the red line indicates the position of detected user 1, and the blue line indicates the position of detected user 2. In Fig. 6b, the green line indicates the detected center position of ridge. Figure 6d–f shows the time history of velocity command. In addition, the light-red highlights in Fig. 6 indicate the time period where the robot recognized user 1 standing. In addition, the light-blue highlights in Fig. 6 indicate the time period where the robot recognized user 2 standing.



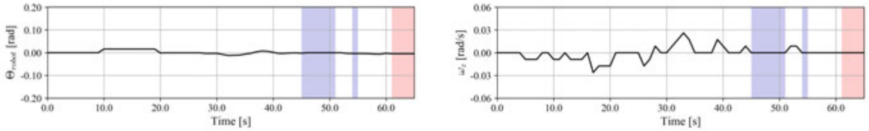
(a) Robot and user positions X in field coordinate system

(d) Velocity command v_x



(b) Robot, ridge and user positions Y in field coordinate system

(e) Velocity command v_y



(c) Robot yaw angle Θ in field coordinate system

(f) Velocity command ω_z

Fig. 6 Time history of robot position and velocity command

As shown in Fig. 5 at time $t = 50.0$, user 2 stood up and performed containers loading. At the time as shown in Fig. 5, the height of the convex MF μ_{state}, considering the user's behavior was 0.0. Then, it was confirmed that the velocity command was calculated to be (0.0, 0.0, 0.0) and the robot had stopped and waited until the user sat down and moved, as shown in Fig. 6d–f.

In addition, as shown in Fig. 6b, e, f, it was confirmed that the robot could track ridge without collision, by adjusting the velocity command v_y, ω_z according to the detected ridge position even where the furrow was quite narrow with respect to the width of the wheel shown in Fig. 1b.

5 Conclusions

In order to realize smooth harvesting support, a ridge-tracking according to the user's behavior, such as movement, harvesting, and loading the yield crops is required. In this study, the ridge-tracking control framework based on fuzzy set theory which can evaluate and integrate multiple situations was proposed. The proposed method was applied to strawberry harvesting support, and experiments were carried out at a strawberry farm using a compact multi-functional agricultural robot "MY DONKEY®". From the experimental results, it was verified that the robot could track the ridge without collision according to the user's behavior by the proposed method.

Acknowledgements his study was supported by "A Framework PRINTEPS to Develop Practical Artificial Intelligence" of the Core Research for Evolutional Science and Technology (CREST) of the Japan Science and Technology Agency (JST) under Grant Number JPMJCR14E3. The experimental fields were provided by The Japan Research Institute, Ltd.

References

1. Bechar, A., Vigneault, C.: Agricultural robots for field operations: concepts and components. Biosyst. Eng. **149**, 94–111 (2016)
2. Shamshiri, R.R., Weltzien, C., Hameed, I.A., Yule, I.J., Grift, T.E., Balasundram, S.K., Pitonakova, L., Ahmad, D., Chowdhary, G.: Research and development in agricultural robotics: a perspective of digital farming. Int. J. Agric. Biol. Eng. **11**(4), 1–14 (2018)
3. Wolfert, S., Ge, L., Verdouw, C., Bogaardt, M.J.: Big data in smart farming–a review. Agric. Syst. **153**, 69–80 (2017)
4. King, A.: The future of agriculture. Nature **544**(7651), S21–S23 (2017)
5. Imperoli, M., Potena, C., Nardi, D., Grisetti, G., Pretto, A.: An effective multi-cue positioning system for agricultural robotics. IEEE Robot. Autom. Lett. **3**(4), 3685–3692 (2018). https://doi.org/10.1109/LRA.2018.2855052
6. Miwa, Y., Ikuma, K., Kidohshi, H.: Era of next-generation agriculture 4.0 that IoT opens (in Japanese). The Nikkan Kogyo Shimbun, Tokyo (2016)

7. The Japan Research Institute, Ltd.: Establishment of next-generation agricultural robot "DONKEY" development consortium (in Japanese). https://www.jri.co.jp/page.jsp?id=31985. Accessed 18 Mar 2019
8. Tsuzaki, R., Yoshida, K.: Motion control based on fuzzy potential method for autonomous mobile robot with omnidirectional vision (in Japanese). J. Robot. Soc. Jpn. **21**(6), 656–662 (2003)

ANYmal in the Field: Solving Industrial Inspection of an Offshore HVDC Platform with a Quadrupedal Robot

C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter

1 Introduction

Offshore wind energy is one of the fastest-growing energy sources in the world, but the exploitation of the renewable energy imposes a number of technical challenges. To transport offshore wind energy to the coast over more than 200 km efficiently, the AC current from wind turbines needs to be converted to DC current on a so-called offshore High Voltage Direct Current (HVDC) converter platform. These platforms are built of a complex integration of various electrical and mechanical equipment such as pumps, water treatment systems, and several safety and backup systems. All of these components can potentially fail and lead to a shutdown of the converter station, which is extremely costly for the operator. To provide electricity to millions of people reliably, such offshore converter stations are regularly inspected by human operators. Due to high expenses for trained personal, transportation, and supplementary infrastructure required, manned platforms are highly undesirable for operators of such platforms. Health and safety considerations of a manned operation lead to the same conclusion. Modern platforms are thus equipped with various digital sensors which are supervised from onshore. However, additional usage of a mobile robot for inspection is desirable for three main reasons: First, the monitoring capability of the robot is very similar (and even better in certain aspects) in comparison to the capacity of humans and significantly favorable compared to fixed installed cameras. The likelihood that a camera, mounted at a fixed position, provides an insufficient perspective for a situational judgement is high. Moreover, a fixed setup is often useless in case of a major incident due to lost communication or damaged equipment. Installation costs and operating expenses of fixed sensors are high, especially for retrofitting older existing sites. A second advantage of the mobile system is that issues can be detected earlier due to more frequent visits of the robot compared to manned inspec-

---

C. Gehring (✉) · P. Fankhauser · L. Isler · R. Diethelm · S. Bachmann · M. Hutter
ANYbotics AG, Weinbergstrasse 35, 8092 Zurich, Switzerland
e-mail: cgehring@anybotics.com

M. Potz · L. Gerstenberg
TenneT Offshore GmbH, Eisenbahnlängsweg 2a, 31275 Lehrte, Germany
e-mail: marcel.potz@tennet.eu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_18

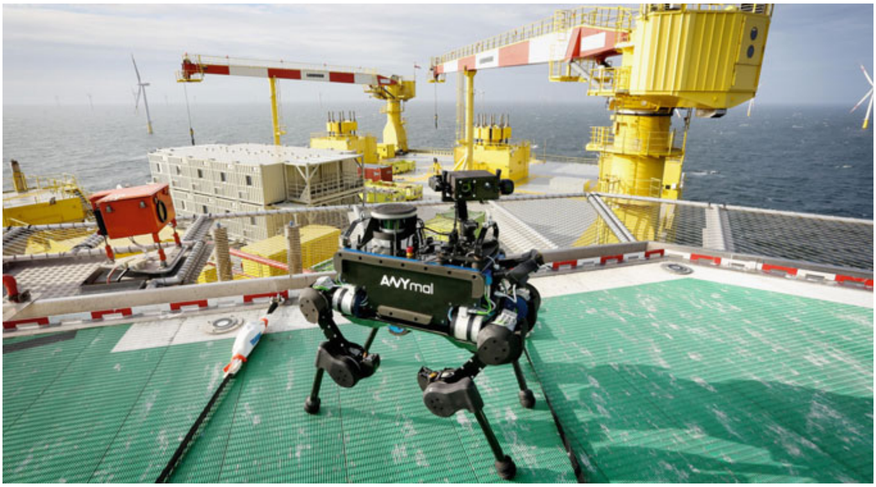


Fig. 1 The quadrupedal robot ANYmal was deployed for autonomous inspection and surveillance on a wind energy HVDC converter platform in the North Sea

tion routines. Furthermore, modern sensor technology allows the mobile system to perceive conditions that are not perceptible for humans. The robot can detect leakages, hot spots, gas leaks, and deteriorating machines before a major problem occurs due to the high accuracy and repeatability of its measurements. Hence, the risk of consequential damage and costs can be reduced substantially. The robot can inspect a problem from different viewpoints and the right specialist for repairing can be sent from the beginning—saving additional costs. The third benefit is the automated collection of vast sensory information. For instance, 3D mapping of the environment is a by-product of the robot's navigation system, but is also highly valuable for the operator. Ultimately, predictive maintenance based on the gathered data can reduce downtime of the site and optimize capital expenditure for backup equipment.

To this date, no robots have been used on offshore HVDC converter stations for surveillance and inspection. One of the main underlying reasons is that these platforms were mainly designed for humans and pose a challenging setting for state-of-the-art mobile robots. In this paper, we present the first findings from extensive field tests for the industrial inspection and surveillance with an autonomous mobile robot on an offshore platform. The tests were performed with ANYbotics' quadrupedal robot ANYmal, which is designed to navigate through difficult environments with the superiority of its legged locomotion in comparison to wheeled or tracked actuation. The deployment took place on one of TenneT's offshore HVDC platforms in the North Sea in 2018 as shown in Fig. 1.

2 Related Work

Due to the complexity of deploying a mobile robot on an offshore platform, only a few attempts have been reported so far. In 2011, [9] conducted a feasibility study with the mobile platform MIMROex on a shallow water offshore gas platform in the South Chinese Sea. The system was programmed to automatically follow a pre-defined path with help of reflective poles for localization and take readings of gauges and gas concentrations. With its four-wheeled design, the robot was restricted to flat ground with a maximum of 20 mm bumps. Similarly, the Sensabot robot [5] was shown to perform remote-controlled inspection task on an onshore oil and gas facility. To overcome the limited mobility of the wheeled actuation, a customized elevator and cog rail ladder were proposed. Another approach to mobility in robotic industrial inspection was presented by [1], which proposed to use a rail-guided system where the robot would travel along a metal tube.

In 2014, the company Total and ANR initiated the ARGOS Challenge (Autonomous Robot for Gas and Oil Sites) which required the systems to autonomously perform various inspection tasks and navigate over difficult terrain including stairs [7]. Out of the five approved teams, four robots were built as tracked vehicles with movable flippers (e.g., [6, 8]), while our group participated with the legged robot ANYmal [4]. These demonstrations showed that autonomous industrial inspection for offshore plants is feasible without requiring to adapt to the environment or install special equipment.

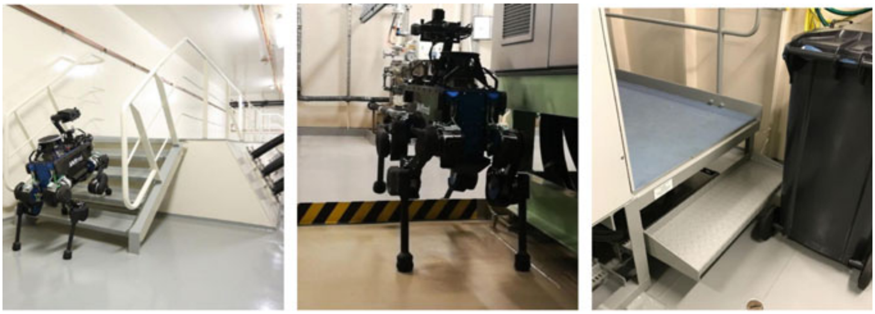
3 Problem Statement

The environment of the HVDC platform and its difficulties for mobile robots as well as the inspection tasks are outlined in this section.

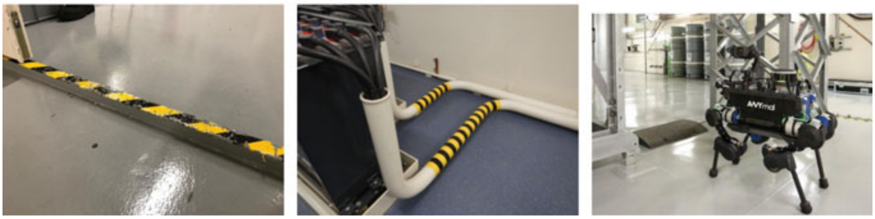
3.1 Description of the Environment

The working area on the platform has an overall length of 80 m, width of about 60 m, and a height of 30 m. The decks of the platform are connected by staircases and an elevator. About 50% of the site cannot be accessed by humans or the robot during operation due to potential electrical discharges. These rooms can only be accessed once per year when the site is shut down for revision. During normal operation of the site, 50 of the ca. 200 rooms of the platform including GIS, transformer, pump, water treatment, AC&DC, fire-fighting, and battery rooms are interesting for robotic inspection.

Most rooms and corridors of the platform have flat ground, but there are numerous areas where the terrain is difficult to overcome by most wheeled or tracked systems.



(a) Stairs (b) Leakage protection (c) Step up



(d) Door sill (e) Pipes (f) Cable protection

Fig. 2 Even though most of the ground is flat, various obstacles on the ground makes the locomotion of the robot difficult

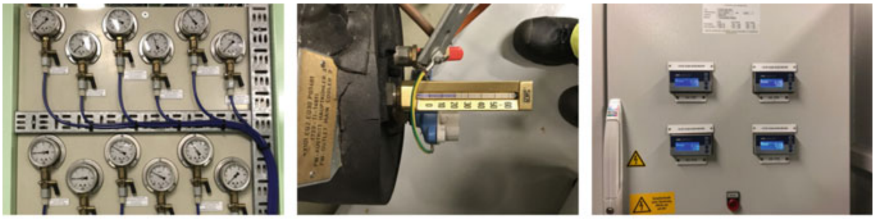
Figure 2 shows some examples of obstacles that can be expected on the platform. In addition, although the platform is usually clean and tidy, the pathway could be obstructed by installed equipment or movable objects.

A further challenge for a mobile robot is the doors of the rooms, which are usually closed. For some doors, even a badge is required to open them. Some of the doors facing outdoors are opening and closing automatically. The seven lower decks of the platform are connected by four indoor and two outdoor staircases as well as an elevator. Some areas of the platform relevant for inspection are only reachable by stairs.

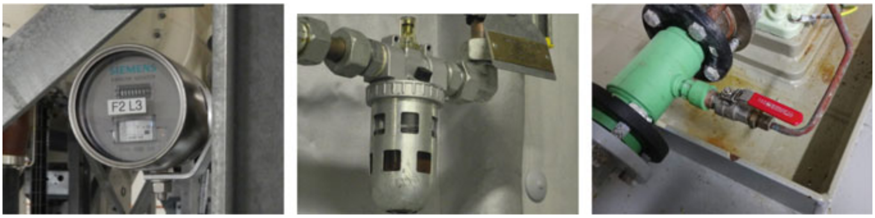
The floors of the rooms with machinery are made of painted metal while other rooms are covered by linoleum. Some corridors and stairs are built with checker boards and gratings. Water, oil, and fine dust can be expected on the ground which can thus become very slippery.

Typical room temperature is 25 °C, but some rooms are not isolated and thus could reach temperatures of 5 °C. Other rooms like the Diesel generator room can become rather hot with 35–45 °C. The rooms are air conditioned with low relative humidity in normal operation. In case of a failure of equipment, it can become high. On the top deck, precipitation can be expected.

The communication between offshore and onshore is enabled by a fiber cable. For continuous monitoring or tele-operating the robot, a reliable wireless network needs to be installed. No wireless network is available in the working areas of the platform. For this reason, WiFi routers were installed for the feasibility tests.



(a) Manometers (b) Thermometer (c) Digital displays



(d) Analog counters (e) Oil level (f) Valve

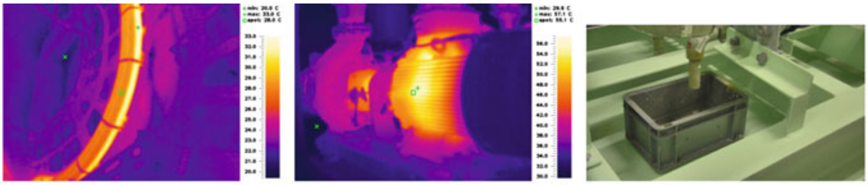
Fig. 3 Various instruments need to be read out to monitor the equipment on the platform

3.2 Robot's Mission

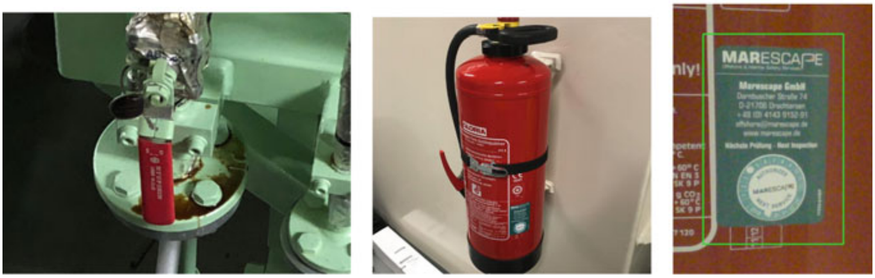
Daily surveillance tours of the robot included visits of specific points of the plant. During these missions, the robot's tasks were reading instruments, checking the health of the equipment of the site, monitoring environmental changes, and detecting anomalies.

On a HVDC platform, there are numerous instruments which are usually read out by technicians. Figure 3 showcases some typical instruments like manometers, oil levels, valves, displays, and panels. Using cameras and computer vision algorithms, the images can be processed automatically and the measured values can be reported with a confidence level indicating how good the recording and interpretation was.

While instruments provide some information about the operational state and health of the machinery, the equipment can be further examined by the robot using thermal imaging, visual inspection, and sound assessment as depicted in Fig. 4. The thermal camera allows to estimate the temperature of cable trails, pumps, and electrical components. The optical camera enables screening the environment for oil leakages. Malfunctioning pumps and fans as well as gas leakages can be early detected based on sound measurements with the onboard microphone. Another typical task of the human service team is to review the expiry date of the fire extinguishers (Fig. 4e). The robot can take pictures of the fire extinguisher and extract the label with the expiry date and highlight it as shown in Fig. 4f. This can be done with generic visual inspection points. If the robot does not find the label or something it was trained to look at, it can inform the remote operator who can take over and have a closer look by tele-operating the robot.

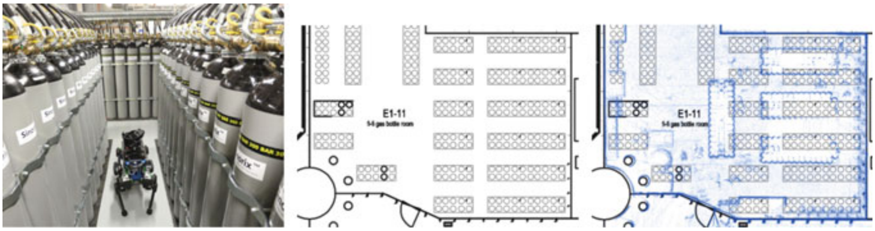


(a) Cables (b) Pump (c) Fluid container



(d) Oil leakage (e) Expire date of fire extinguisher (f) Indicated label

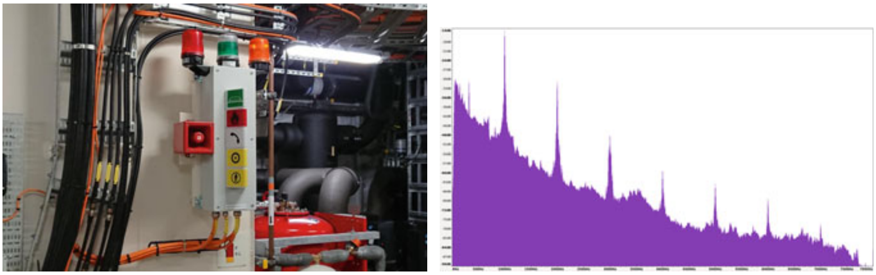
Fig. 4 The health of the equipment is monitored by thermal imaging, visual inspection, and sound assessment



(a) Actual room (b) Floor plan (c) Recorded map

Fig. 5 The bottles (**a**) have been placed differently as indicated by the floor plan (**b**). The blue points recorded by the robot's laser sensor show the correct map (**c**)

While construction or maintenance work is ongoing on the platform, the environment can change. Since the robot maps the environment with its sensors for navigation, the same data can be used for visualizing these changes. The bottles in the room shown in Fig. 5a are arranged differently as illustrated in the floor plan depicted in Fig. 5b. The 3D points in blue measured by the robot's laser sensor drawn in Fig. 5c visualizes the environmental changes. The sensory information of the robot can also be used to check if any escape routes are blocked or equipment is missing. Additionally, events and anomalies can be detected by the robot. For instance, the ambient temperature can be monitored to detect fire. With the microphone, the robot can also check if the general platform alarm shown in Fig. 6a is operational as expected by analyzing the frequency response as depicted in Fig. 6b.



(a) General platform alarm (b) Frequency analysis

Fig. 6 The robot can record the general platform alarm (**a**) with its microphone and detect the alarm according to the identified frequency peaks (**b**)

4 The Robotic Inspection System

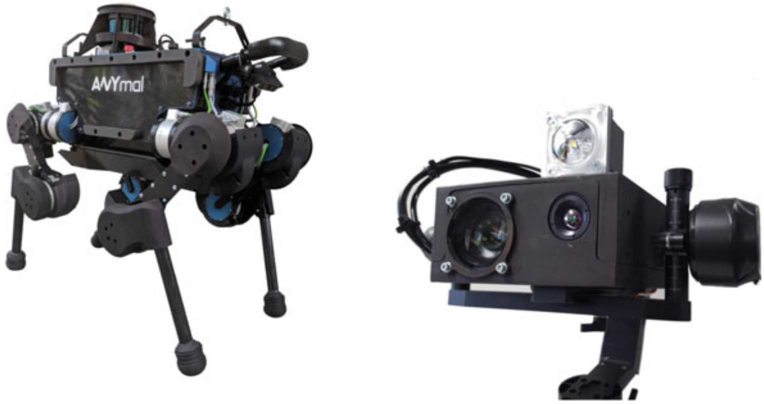
The robot system which was used during the field tests on the HVDC platform is a newer version of the quadrupedal robot ANYmal as described in [3, 4] with stronger actuators for climbing steep stairs up to 40 ^{c}irc. The ANYmal is an electrically driven quadrupedal robot designed as a general-purpose platform. With the size of 800 × 600 × 700 mm in standing configuration, the weight of about 30 kg, and a payload capability of 10 kg, the robot is well suited for unmanned inspection tasks. Onboard computers provide power for complex optimization and vision tasks, and the battery ensures about 2–4 h autonomous operation. The system is completely sealed against dust and water ingress (IP67).

4.1 Sensors and Actuators

Numerous sensors are mounted on the robot for navigation, inspection, and diagnostics. The robot is able to post-process sensor measurements onboard. This allows the robot to operate even in areas with limited connectivity.

For inspection of the offshore site, it was carrying a prototype sensor payload for inspection as shown in Fig. 7. An actuated gimbal with a visual and thermal camera as well as a flashlight are mounted on the top of the robot. The actuated gimbal increases the workspace of the inspection sensors, because the cameras can be aligned to the inspection goals independent of the robot's posture. This allows inspection of the ceiling right above the robot. The robot features microphones for audible and ultrasonic sound recording of machineries like pumps or fans.

To navigate through the environment, the robot uses perception sensors to localize itself in the environment and to avoid obstacles. For far-field perception up to 100 m, a Velodyne Puck LIDAR sensor with 16 lasers beams creates a point cloud of the environment at 10 Hz to map the environment and localize the robot. For near-field



(a) ANYmal (b) Interim Inspection Payload

Fig. 7 The robot ANYmal B300 (**a**) together with the prototype inspection payload (**b**) has been used for the field tests

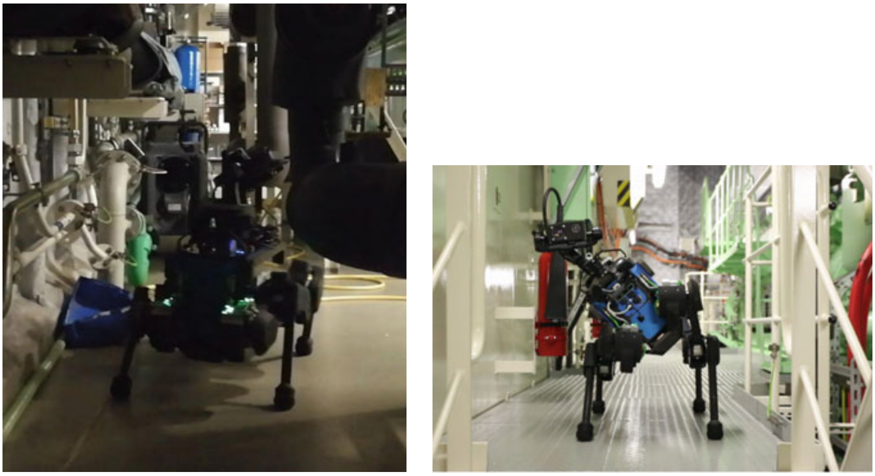
perception up to 7 m, an assisted stereo camera, the Intel RealSense D435, records depth information around the robot. The dense information of the ground is used for accurate foothold planning and obstacle avoidance.

4.2 Mobility and Navigation

The robot fuses the 3D point cloud information from the LIDAR sensor and depth camera for mapping and localization. Employing an ICP-based localization method as described in [4], the robot determines its current position and orientation with respect to a previously recorded map. A pose graph is used as a map representation to roughly define where the robot can move within the 3D point cloud of the environment. The robot's pose of the navigation goals, via-points and checkpoints are stored in such a pose graph as illustrated in the user interface in Fig. 9. The poses are inter-connected with edges (green lines) to indicate that the robot can move between these poses. When a node of the pose graph is selected as a goal, the robot searches the pose graph for the shortest path from its current location to the target pose using an A* algorithm.

Knowing the path from the pose graph, the robot plans a smooth trajectory from its current position along that path. The trajectory is generated online using the robot's perception sensors such that obstacles are automatically avoided.

Depending on the terrain and speed, the robot uses different gaits for locomotion. The robot employs a trotting gait on smooth and flat terrain for fast walking up to 1.0 m/s. In more challenging terrain, the robot makes use of a slower walking gait as depicted in Fig. 8a, which even enables crawling underneath suspended obstacles.



(a) Crawling through narrow corridors (b) Pose adaptation for inspection

Fig. 8 ANYmal employs different gaits for locomotion and can adapt its posture for optimal inspection

To overcome unknown obstacles, the robot relies on its perception capabilities to safely plan its footholds and adapt its body pose [2]. If obstacles are known like steps, special maneuvers can be also pre-programmed. To inspect a checkpoint, the robot walks to a defined viewpoint in the 3D map and adapts its posture if required as shown in Fig. 8b.

4.3 Robot Control and User Interface

The operator commands the robot via WiFi link. For teleoperation, the robot is controlled by the graphical user interface (GUI) shown in Fig. 9. The GUI visualizes the robot in 3D together with the environment, planned path and inspection points. The state of the robot including the health of all sensors and actuators as well as the charge of the battery is shown on the front page. Scheduled tasks and progress of an ongoing inspection tour can be observed remotely. For deployment of the robot on a new site, a remote control is additionally provided to steer the robot.

5 Field Tests

Autonomous inspection tours similar to the ones performed by human operators have been evaluated on the offshore platform. A mission usually starts and ends on the

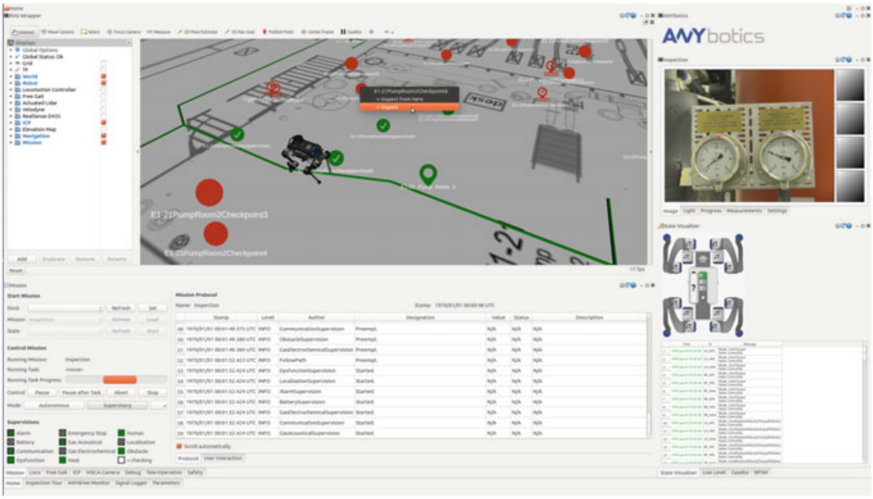
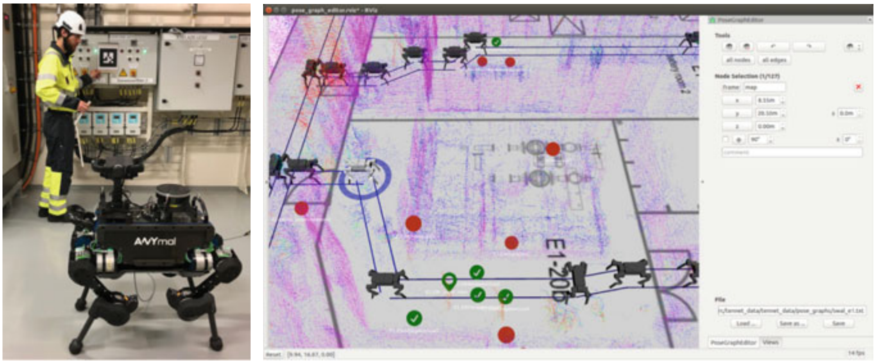


Fig. 9 The Graphical User Interface (GUI) shows the state of the robot, the measurements of the onboard sensors, and the progress of the ongoing inspection tour

charging station. Missions are easily programmable by a qualified operator through the graphical user interface described in Fig. 4.3. The operator can access the status of the system and the progress of the mission at any time and automatically receives a post-mission report as soon as the mission is completed. The report includes information about the state of the system, locomotion, localization, navigation, inspection, and unexpected events. During an autonomous mission, the operator is able to take direct remote control of the robot at any time. This enables the operator to control the robot in difficult situations or send it to any point on the map to have a closer look at its environment.

To conduct any field tests on the offshore platform, the field engineers needed to successfully pass safety training. After installation of the WiFi network and charging system on the platform, functional tests of the robot's localization and navigation system were conducted before the robot was sent on fully autonomous missions. The deployment of the system in a new environment required the following steps: First, the robot is manually steered through the environment using a remote control. While scanning the environment with the LIDAR sensor and depth camera, the robot automatically builds a 3D map of its environment. The recorded point cloud is then stored on the robot and can be inspected by the operator in the GUI. In a second round, the robot is steered to the points of interest. The operator holds an AR tag next to the checkpoint as shown in Fig. 10a. The location of the robot together with the location of the checkpoint is computed automatically as the robot knows its location with respect to the previously recorded map, and the pattern of the AR tag provides 3D pose information of the checkpoint relative to the onboard camera of the robot. The operator defines the type of checkpoint and programs its characteristics



(a) Checkpoint Recording

(b) Mission Editor

Fig. 10 The location of an inspection point (**a**) is recorded by holding an AR tag next to the point of interest. After recording the checkpoints, the path of the robot can be adapted in a GUI (**b**)

such as pressure range for a manometer or maximum temperature of a hot spot. For navigation, the checkpoints and via-points are defined in the pose graph. An editor is available to modify and adapt the path of the robot in 3D for the mission as shown in Fig. 10b. The robot's reactions to unexpected events such as communication loss are programmed in the next step. The robot, for instance, waits for further instructions from the operator when it found an unknown obstacle or goes back to the starting point when communication is lost. A physical simulation of the robot is provided to thoroughly test navigation and inspection before deploying it on the real system. Errors made by the operator such as unconnected locations can be simply found before costly experiments are conducted. Once the mission is configured and tested in simulation, tests on the actual site are made to check the reliability of navigation and inspection.

6 Results of the Feasibility Study

During the field tests of two weeks, ANYmal was executing more than 30 autonomous missions. The robot's path and the 19 points of interest of one of the autonomous missions are illustrated on the floor plan in Fig. 11. The inspection tour took about 25 m including 6 m for inspection on average.

The coverage of points of interest by ANYmal has been analyzed in a pump room, water treatment room, and HVA/C room. 385 instruments like manometers, thermometers, valves, displays, panels, flow meters, pump sound, and pneumatic lubricators have been evaluated in these rooms. While 92 and 85% of the points of interest can be successfully inspected in the pump and water treatment room, respectively, 54% can be scanned by the robot in HVA/C room. Many valves in the HVA/C room are very difficult to see because they are hidden and facing the wall.

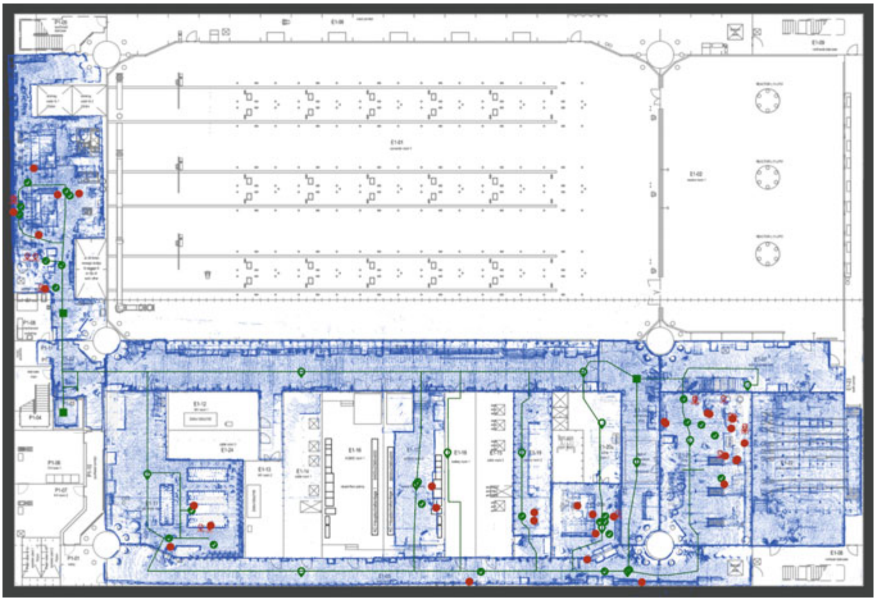
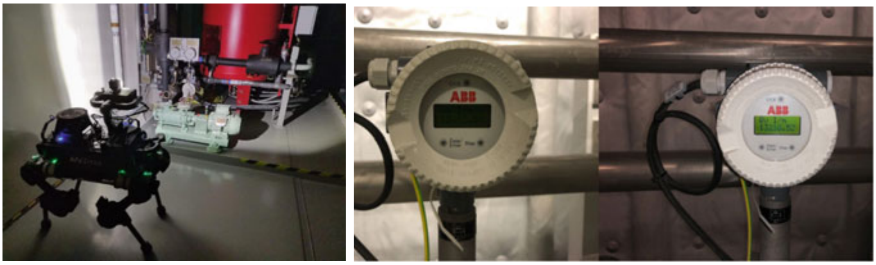


Fig. 11 The path (green) and inspection points (red) of the autonomous inspection mission is shown on the floor plan with the overlayed recorded map of the robot's sensors (blue dots)

The rooms with installed equipment typically have no windows and thus no natural light. However, all rooms of the platform are illuminated by lamps. By default, about one-third of the lamps are always switched on for emergency situations. The remaining lamps can be switched on by pushing a toggle switch which could be used by the robot with additional manipulation skills. Figure 12a shows the robot inspecting a room in complete darkness with its onboard flashlight. Tests showed that the onboard light can improve visual inspection as shown in Fig. 12b. While the illumination is usually good enough for visual inspection if all lamps are on, the noise level of images becomes critical when the emergency light is only available. In conclusion, an onboard flashlight is recommended for inspection.

In addition to the visual inspection tasks, acoustic and thermal inspection was successfully tested. The alarm for emergency situations of the siren shown in Fig. 6a was successfully identified by the robot's microphone and onboard frequency analysis. Thermal images of cables and pumps as depicted in Fig. 4a were automatically recorded by the robot during its inspection tours.



(a) Inspection in the darkness

(b) Inspection with and without flashlight

Fig. 12 In case the lights are off as in **a**, the robot can use its onboard flash light for inspection. The onboard flashlight also improves the inspection when the lights are switched on **b**

7 Conclusion

The presented study addressed the feasibility of using a legged robot for automated and unmanned inspection tasks on the offshore platform. The mobility and inspection skills of the robot ANYmal were demonstrated on a HVDC converter station. For illustration, we collected a video summary of the field tests.[1] In conclusion, the quadrupedal robot ANYmal is well suited for autonomous inspection of a HVDC platform. The robot can overcome various obstacles with its legs, pass through narrow passages due to its slim design, and can observe the environment from a visual, thermal, auditory, and geometrical perspective. The only major impediment to solve is how to open and pass doors. Due to safety considerations regarding fire, not all doors could be automated in both directions. Apart from opening doors, the robot could also use a manipulator to push buttons, toggle switches and fuses, and turn valves. Integrating the inspection sensors in a robotic arm would further increase the coverage of points of interest for inspection.

While human operators inspect the platform monthly, the robot can survey it more often, even daily, in the future. To make this happen, the integration of the robot's software into the existing control system of the platform and the long-distance communication need to be addressed in the future.

References

1. Carvalho, G.P., Xaud, M.F., Marcovistz, I., Neves, A.F., Costa, R.R.: The DORIS offshore robot: recent developments and real-world demonstration results. In: Proceedings of IFAC. Elsevier B.V. (2017)
2. Fankhauser, P.: Perceptive locomotion for legged robots in rough terrain. Doctoral Thesis. ETH Zurich (2018)

---

[1] https://youtu.be/DzTvIPrt0DY.

3. Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C.D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Blsch, M., Kolvenbach, H., Bjelonic, M., Isler, L., Meyer, K.: ANYmal—toward legged robots for harsh environments. Adv. Robot. **31**(17), 918–931 (2017). ISSN: 0169-1864
4. Hutter, M., Diethelm, R., Bachmann, S., Fankhauser, P., Gehring, C., Tsounis, V., Lauber, A., Guenther, F., Bjelonic, M., Isler, L., Kolvenbach, H., Meyer, K., Hoepflinger, M.: Towards a generic solution for inspection of industrial sites (2017)
5. JPT Staff: Sensabot: a safe and cost-effective inspection solution. J. Pet. Technol. **64**(10), 32–34 (2015)
6. Kohlbrecher, S., von Stryk, O.: From RoboCup rescue to supervised autonomous mobile robots for remote inspection of industrial plants. KI Künstliche Intelligenz **30**(3-4), 311–314 (2016)
7. Kydd, K., Macrez, S., Pourcel, P.: Autonomous robot for gas and oil sites. In: SPE Offshore Europe Conference and Exhibition (2015)
8. Nagatani, K., Endo, D., Watanabe, A., Koyanagi, E.: Design and development of explosion-proof tracked vehicle for inspection of offshore oil plant. In: Conference on Field and Service Robotics (FSR), pp. 531–544 (2017)
9. Pfeiffer, K., Bengel, M., Bubeck, A.: Offshore robotics—survey, implementation, outlook. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 241–246. IEEE (2011)

Large-Scale 3D Mapping of Subarctic Forests

Philippe Babin, Philippe Dandurand, Vladimír Kubelka, Philippe Giguère, and François Pomerleau

1 Introduction

Autonomous mobile robots require a representation (i.e., a map) of the environment in order to perform specific tasks. For instance, maps are needed internally to plan motions and avoid obstacles. The map itself can also be the objective, captured by robots and used for later analysis [1], including forestry inventories [2]. Although many solutions exist for localization and mapping, the environment itself influences the complexity of the task, and thus dictates which algorithm to use. In this study, we targeted snowy, subarctic forests to explore new challenges to large-scale mapping. Indeed, this type of environment is minimally structured, making registration more difficult. Moreover, the ruggedness of terrains dictates the need for a full six Degrees of Freedom (DoF) solution, with a little assumption on trajectory smoothness.

Another difficulty brought by subarctic environments is the lack of distinctively visual features during snowy periods [3]. With images of snowy surfaces, it is challenging to extract enough features in order to support visual odometry or vision-based Simultaneous Localization and Mapping (SLAM). This precludes the use of passive camera-based localization systems, making lidar the sensing modality of choice for these conditions. A natural approach to mapping in this case is to incrementally build a 3D point cloud map from scans taken at different locations, using the Iterative Closest Point (ICP) algorithm [4]. The ICP algorithm iteratively finds corresponding points between two point clouds and looks for a rigid transformation minimizing an alignment error. This approach ensures local consistency, yet inevitably suffers from global drift [5]. This drift problem can be mitigated by SLAM techniques based on pose graph optimization [6, 7]. The key idea behind the latter is to identify loop closures and optimize all estimated transformations between the individual lidar scans to ensure global consistency. Unfortunately, in environments which do not allow loop closure (such as a long straight trajectory), drift cannot be avoided without additional external localization sources, such as Global Navigation Satellite System

---

P. Babin (✉) · P. Dandurand · V. Kubelka · P. Giguère · F. Pomerleau
Northern Robotics Laboratory, Université Laval, Quebec City, Canada
e-mail: philippe.babin.1@ulaval.ca

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_19

(GNSS). Moreover, the pose graph optimizer requires uncertainty estimations in the form of covariance matrices for all transformations, including those from ICP. As shown by [8], this uncertainty can be modeled, learned, or sampled, but the Gaussian distribution assumption does not hold up well in complex 3D environments.

On the other hand, autonomous robots operating on polar ice sheets [9] often rely on GNSS as their main source of positioning. In unstructured environments (e.g., boreal forests and taiga), GNSS cannot be used this way due to the high uncertainty of position estimates. This uncertainty is caused by interference of the canopy with the signals from satellites [10]. Still, the main advantage of GNSS is that it provides a global source of positioning, which shows minimal and bounded bias compared to the Iterative Closest Point (ICP). Meanwhile, ICP creates maps that are crisp (i.e., locally consistent).

The goal of our paper is to demonstrate large-scale mapping of difficult environments, while generating maps that are (i) crisp, (ii) without long-term drifts, and (iii) that can be updated swiftly. To satisfy the first two criteria, we experimented with embedding external constraints directly within the Iterative Closest Point (ICP) minimization process using a novel formulation to handle uncertainty on positions. More precisely, we propose to augment the ICP algorithm by adding penalty terms based on the global GNSS positioning and the Inertial Measurement Unit (IMU) attitude estimates, weighed by their uncertainty. This formulation has the advantage that the uncertainty associated with these external constraints, contrary to the ICP's, can usually be readily estimated. Second, it avoids undue oscillations induced by alternating between graph minimization and point cloud registration, as both algorithms have no guarantees of sharing the same minimum. The third criterion pertains to the need for fast point-cloud map update in autonomous systems. The main bottleneck of the ICP algorithm is the update of the KD-tree, a data structure used for a fast nearest neighbor search. Since our objective is to build large maps, this slowdown becomes unacceptable. We investigate a simple optimization technique to reduce the execution time of registrations by reducing the size of the map portion used for the KD-tree update. We tested our approach on large-scale maps recorded in a boreal forest during the winter (shown in Fig. 1). A particular emphasis has been placed on discussing problems related to the different aspects of lidar mapping for this type of environment.

2 Related Work

The context of our work involves robotic deployments in harsh, snowy environments. This problem of mapping and localization in these conditions has only been investigated by a few publications. For example, visual SLAM for a robotic snowmobile platform was deployed by [11] on a glacier in Alaska. The authors report difficulties relating to the relatively low number of visual features in close vicinity of the mapping platform, compared to visual features located on the horizon. Since the nearby features are vital for translation estimation, image processing techniques

Fig. 1 The boreal forest *Forêt Montmorency* presents an environment ideal to test robotic applications in a subarctic region. For lidar mapping, it offers challenging dense forest conditions. *Credit Forêt Montmorency*

are proposed to improve their extraction. The effects of changing shape and appearance of snowy areas on a path-following algorithm are also discussed in [12]. Their findings further motivate the use of lidars for mapping in these conditions. In our approach, the deployment of the lidar sensor translates the problem of extracting image features for localization into the problem of locating against 3D geometry. Areas covered by the boreal forests comply well with this requirement. On the other hand, on open plateaus where 3D features are sparse, the GNSS constraints assure consistent localization and mapping. Moreover, similar to [11, 12], we do not require wheel or track odometry measurements from the mobile platform. This feature simplifies the integration of the mapping system into different mobile vehicles which do not offer any odometry (in our case, the sleigh). We, however, benefit from an IMU which provides attitude prior to unbiased roll and pitch angles.

Laser scans can be captured from a ground-based static sensor as in the work of [13] who employed a stationary high-density lidar sensor for terrain classification and identification of tree stems. Alternatively, the Airborne Lidar Scanning (ALS) approach allows the mapping of vast forested areas from the air. Besides the ALS, the *Structure from Motion* technique [14] and stereo imagery [15] are further alternatives to creating 3D maps, suitable mainly for light aerial drones. Our goal is to create and maintain globally consistent 3D maps for autonomous ground robots. In the case of ALS, global consistency is easier to achieve because of the high-altitude point of view and unobstructed GNSS reception. Contrarily, ground robots only observe a limited portion of the area at a time and their GNSS reception is partially occluded by the canopy. On the other hand, ground-based 3D scans offer high details, also useful for in-depth vegetation analysis. The problem of storing and managing large amounts of data is common to all of the mentioned works. In our approach, we propose a technique to limit the computation demands during the mapping process.

Fueled by the increasing interest in self-driving cars, multiple large-scale urban datasets, most notably [16], (containing lidar and GNSS information besides other sensors) have become available. These datasets have accelerated development, refining a variety of visual- and lidar-based SLAM algorithms. Contrary to structured urban environments, we investigate the characteristics of mapping in unstructured ones (forests) in harsh winter conditions. Additionally, any improvement in the accuracy of registration algorithms reduces pressure on loop-closure and graph minimization algorithms, leading to more robust lidar-based SLAM algorithms overall.

From the extensive family of Iterative Closest Point (ICP) variants [17], our contribution relates mainly to incorporating generalized noise models into the ICP algorithm. Since the GNSS positioning provides a confidence estimate in the form of a covariance matrix, simplification to an isotropic noise model discards potentially important information. [18] were the first to consider anisotropic and inhomogeneous noise models when estimating optimal rotation of features extracted from stereo-pair depth images for 3D reconstruction. Later, the Generalized Total-Least-Squares ICP (GTLS-ICP) algorithm was introduced by [19] for registering medical fiduciary markers. The work considers an anisotropic noise model in the registration phase of ICP and accounts for optimizing the translation component as well. Further improvements were introduced by [20], where the matching phase is modified to benefit both from KD-tree search speed and Mahalanobis-distance metric. This technique has been eventually enhanced by the introduction of a new kind of KD-tree which directly supports Mahalanobis distance and a new minimizer [21]. In our approach, the anisotropic noise is strictly limited to the GNSS position measurements, making the problem slightly different. We look for a way to integrate this positioning information together with its anisotropic noise into the ICP. More closely related to robotic applications, the Generalized-ICP (GICP) algorithm [22] preserves the possibility to model measurement noise, while focusing on minimizing the plane-to-plane metric. However, using an iterative minimizer such as Broyden–Fletcher-Goldfarb–Shanno (BFGS) inside the matching loop of ICP is prohibitively slow. In this paper, we investigate how to link point set registration to include penalty terms brought by the GNSS and IMU measurements within the same mathematical framework generic to anisotropic noise.

3 Theory

3.1 New Formulation for Point-to-Gaussian Cost Function

The Iterative Closest Point (ICP) algorithm aims at estimating a rigid transformation $\widehat{\boldsymbol{T}}$ that best aligns a set of 3D points $\mathcal{Q}$ (i.e., a *map* point cloud) with a second set of 3D points $\mathcal{P}$ (i.e., *scan* point cloud), given a prior transformation $\check{\boldsymbol{T}}$. For a better representation of surfaces, the points of the map point cloud can be represented locally by planes. The problem of rigid registration using points from the scan and planes from the map [23] can be summarized as minimizing the point-to-plane cost function $\mathrm{J}_{\text{p-n}}(\cdot)$ following

$$\widehat{\boldsymbol{T}} = \arg\min_{\boldsymbol{T}} \; J_{\text{p-n}}\left(\mathcal{Q}, \mathcal{P}, \check{\boldsymbol{T}}\right), \text{ with} \tag{1}$$

$$J_{\text{p-n}} = \sum_{i=1} \sum_{j=1} w_{ij} (\boldsymbol{e}_{ij}^T \boldsymbol{n}_i)^2, \text{ and } \boldsymbol{e}_{ij} = \boldsymbol{q}_i - \check{\boldsymbol{R}} \boldsymbol{p}_j - \check{\boldsymbol{t}}, \tag{2}$$

where $\boldsymbol{e}_{ij}$ is the error vector between the ith point $\boldsymbol{q}$ of $\mathcal{Q}$ and the jth point $\boldsymbol{p}$ of $\mathcal{P}$, $\boldsymbol{n}_i$ is the normal of the plane, $\check{\boldsymbol{R}}$ and $\check{\boldsymbol{t}}$ are, respectively, the rotation and translation part of $\check{\boldsymbol{T}}$, and w_{ij} is a weight limiting the impact of outliers as surveyed by [24]. The double summation in (1) is expensive to compute and is typically approximated using a subset of pairs using the nearest neighbor points of each scan point. To simplify the notation, we will use $\boldsymbol{e}_m$ for each error to be minimized, with m being the index of this subset. Point-to-plane error outperforms point-to-point error in most cases [5]. However, it does not represent a non-planar surface well. Point-to-Gaussian provides a more versatile representation [25]. Instead of being represented by a plane, each point in $\mathcal{Q}$ is the mean of a Gaussian and its incertitude is represented by a covariance. The point-to-Gaussian cost function $\text{J}_{\text{p-g}}$ thus becomes the following:

$$\text{J}_{\text{p-g}} = \sum_{m=1} \left(\boldsymbol{e}^T \boldsymbol{W}^{-1} \boldsymbol{e} \right)_m , \tag{3}$$

where $\boldsymbol{W}^{-1}$ is the inverse of the covariance. In point-to-Gaussian, the Mahalanobis distance is minimized instead of the Euclidean distance (point-to-point) or the projected distance to a plan (point-to-plane) . Instead of using a second iterative solver within the matching loop of Iterative Closest Point ICP [19, 21, 22], we propose a novel decomposition to minimize the point-to-Gaussian error (3) directly using the equations for point-to-plane error (1). The inverse of the covariance $\boldsymbol{W}^{-1}$ can be expressed as a matrix $\boldsymbol{N}$ of eigenvectors and a diagonal matrix $\boldsymbol{\Lambda}$ holding the sorted eigenvalues, with $\lambda_1 < \lambda_2 < \lambda_3$, using

$$\boldsymbol{W} = \boldsymbol{N} \boldsymbol{\Lambda} \boldsymbol{N}^T \Rightarrow \boldsymbol{W}^{-1} = \boldsymbol{N} \boldsymbol{\Lambda}^{-1} \boldsymbol{N}^T .$$

The decomposition can be inserted inside the cost function and reformulated as three point-to-plane errors using a projection for each of the eigenvectors. Dropping the summation and the indices for clarity, we obtain for a single pair of points

$$\begin{aligned} \text{J}_{\text{p-g}} &= \boldsymbol{e}^T \boldsymbol{N} \boldsymbol{\Lambda}^{-1} \boldsymbol{N}^T \boldsymbol{e} && (4) \\ &= \boldsymbol{e}^T \left[\boldsymbol{n}_1 \ \boldsymbol{n}_2 \ \boldsymbol{n}_3 \right] \operatorname{diag} \left(\tfrac{1}{\lambda_1}, \tfrac{1}{\lambda_2}, \tfrac{1}{\lambda_3} \right) \left[\boldsymbol{n}_1 \ \boldsymbol{n}_2 \ \boldsymbol{n}_3 \right]^T \boldsymbol{e} \\ &= \underbrace{\frac{1}{\lambda_1} \left(\boldsymbol{e}^T \boldsymbol{n}_1 \right)^2}_{\text{J}_{\text{p-n}}} + \frac{1}{\lambda_2} \left(\boldsymbol{e}^T \boldsymbol{n}_2 \right)^2 + \frac{1}{\lambda_3} \left(\boldsymbol{e}^T \boldsymbol{n}_3 \right)^2 , && (5) \end{aligned}$$

where λ_i is an eigenvalue and $\boldsymbol{n}_i$ is its associated eigenvector. Thus, point-to-Gaussian can be used with any point-to-plane minimizer. In fact, point-to-plane is a special case of point-to-Gaussian, where the first eigenvalue λ_1 is small enough compared to λ_2 and λ_3. This formulation can be used to also minimize Gaussian-to-Gaussian by setting $\boldsymbol{W}$ to the sum of the uncertainty of point $\boldsymbol{q}$ with the rotated uncertainty of its associated $\boldsymbol{p}$.

3.2 Adding Penalty Terms to ICP

ICP mapping creates crisp maps by taking into account local geometric characteristics contained in each new point cloud. Therefore, global consistency is not enforced. On the other hand, GNSS provides globally consistent positioning, but yields low local precision, especially when compared to ICP in forested areas. Furthermore, there is a disproportion between the altitude, latitude, and longitude positioning components, the altitude being the least precise. By fusing ICP, GNSS, and IMU information, we propose to compensate for the ICP drift.

Penalties are a natural way to add a constraint to the minimization step of ICP. They can be seen as imaginary points added to the point cloud during minimization for which the association is known. The minimization problem thus becomes

$$\widehat{\boldsymbol{T}} = \arg\min_{\boldsymbol{T}} \underbrace{\frac{1}{M}\sum_{m=1}\left(w\boldsymbol{e}^T\boldsymbol{W}^{-1}\boldsymbol{e}\right)_m}_{\text{point clouds}} + \underbrace{\frac{1}{K}\sum_{k=1}\left(\boldsymbol{e}^T\boldsymbol{W}^{-1}\boldsymbol{e}\right)_k}_{\text{penalties}}, \tag{6}$$

where $\boldsymbol{e}_k$ and $\boldsymbol{W}_k$ are, respectively, the error and the covariance of the kth penalty, M is the number of matched points, and K the number of penalty added. The penalty error $\boldsymbol{e}_k$ consists of two points added to their respecting frames of reference

$$\boldsymbol{e}_k =^{\mathrm{M}} \boldsymbol{q}_k -^{\mathrm{M}}_{S} \check{\boldsymbol{T}}^{\mathrm{S}} \boldsymbol{p}_k, \tag{7}$$

where ${}^{\mathrm{M}}\boldsymbol{q}_k$ is the position of the kth penalty in the map frame, ${}^{\mathrm{M}}_{S}\boldsymbol{T}$ is the transformation from the scan frame (S) to the map frame (M) at the current iteration, and ${}^{\mathrm{S}}\boldsymbol{p}_k$ is the position of the kth penalty in the scan frame. For instance for the GNSS penalty, ${}^{\mathrm{M}}\boldsymbol{q}_k$ is the GNSS's global position and ${}^{\mathrm{S}}\boldsymbol{p}_k$ is the origin of the scan frame. The effects of adding penalty points are presented in Sect. 5.1.

Since the GNSS penalty points come in the form of a Gaussian distribution and point clouds contain plane normal information, Eq. (6) is approximated using a constant scale s in our implementation. This constant ensures the conversion from projected distances to Mahalanobis distances by assuming that $\frac{1}{\lambda_1}$ is constant for all points and neglecting $\frac{1}{\lambda_2}$ and $\frac{1}{\lambda_3}$ in (5). In this setting, the final cost function to be optimized becomes

$$\mathrm{J}_{\text{p-g}} \approx \frac{s}{M}\sum_{m=1}\left(w(\boldsymbol{e}^T\boldsymbol{n}_1)^2\right)_m + \frac{1}{K}\sum_{k=1}\left(\boldsymbol{e}^T\boldsymbol{W}^{-1}\boldsymbol{e}\right)_k. \tag{8}$$

3.3 Iterative Closest Point Mapping

The mapping was achieved using a modified version of `ethz-icp-mapping` [26]. The mapper performs the following steps: (1) Move the scan to the initial estimate, (2) register the scan with the map using ICP, and (3) insert the scan inside the map.

The initial estimate $\check{\boldsymbol{T}}$ is composed of a translation increment based on the GNSS positioning and change in orientation based on the IMU. The IMU heading is corrected by the GNSS positioning as long as the platform moves forward. Justification of this correction and a possible alternative are discussed in Sect. 5.1. Since this initial estimate $\check{\boldsymbol{T}}$ is utilized in an incremental manner, the mapping can diverge over time. As for the construction of the global map $\mathcal{Q}$, the whole scan $\mathcal{P}$ is not directly concatenated. Rather, only points that are farther than ε from any points in $\mathcal{Q}$ are inserted. This helps in keeping the global map uniform, without sacrificing registration precision. As the robot explores the environment, the complexity of registration grows linearly with the number of points in the map due to the KD-tree structure updates. To stabilize the mapping complexity, a scan $\mathcal{P}$ is not registered against the whole map $\mathcal{Q}$, but only against a subsection of the map within a radius $r_{\max}$ equal to the maximum range of the lidar. The effects of this optimization are shown in Sect. 5.3.

4 Experimental Setup

4.1 *Data Acquisition Platform*

For our experiments, we developed a rugged data acquisition platform which can withstand snow and sub-zero temperatures. It comprises an Xsens MTI-30 IMU, a RoboSense RS-16 lidar, and a REACH RS+ GNSS antenna powered by two 20 Ah 12 V AGM batteries (10 h battery life). A small, low-power computer (AIV-APL1V1FL-PT1) records the sensor data using the Robotic Operating System (ROS) framework. This platform can be attached to most mobile vehicles (see Fig. 2). The rotation axis of the lidar sensor is at an angle of 27° from the vertical. This orientation has been chosen for two reasons: (1) the lidar does not see the mobile vehicle nor its operators as long as they are in front of the platform, and (2) as mentioned in [27] a lower incidence angle with the ground reduces the odometry drift. The GNSS antenna is coupled with a fixed station (also a REACH RS+ antenna) mounted on a tripod to provide a Real-Time Kinematic (RTK) solution.

Université Laval owns the *Forêt Montmorency*, the largest research forest in the world with over 412 km^2 of boreal forest (Fig. 1). For our experiments, we collected data along three large loops (see Fig. 3). Two of them (i.e., `lake` and `forest`) consist of a mix of narrow walkable trails and wider snowmobile trails, while the last one (i.e., `skidoo`) followed exclusively a wide snowmobile trail.

More specifically, the dataset `lake` was recorded by mounting our platform to a snowmobile (see Fig. 2) and then having two operators pull the sleigh through a pedestrian trail. The snowmobile drove through a cross-country skiing trail, which is an open area with good GNSS coverage (`A` to `B` in Fig. 3). The pedestrian trail was a dense forest path (`B` to `C`), where the platform suffered from poor GNSS reception due to the tree canopy. The overlap between scans diminished abruptly

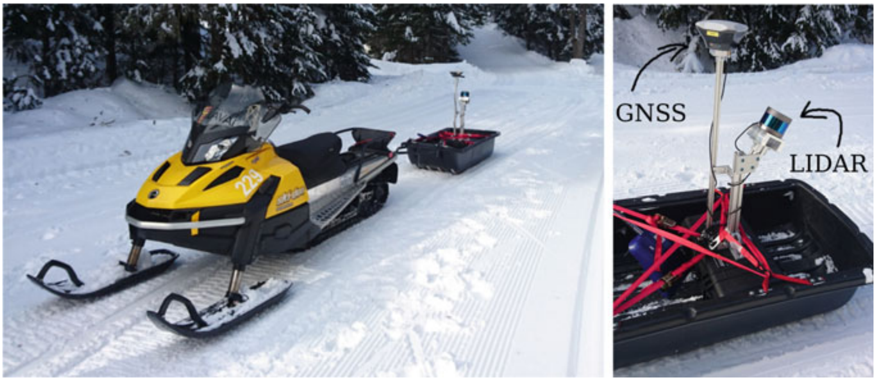


Fig. 2 The data acquisition platform mounted on a sleigh behind the snowmobile (*left*). This configuration limited the transfer of engine vibrations to the sensors. *Right*: close-up of the GNSS receiver and the RS-16 lidar

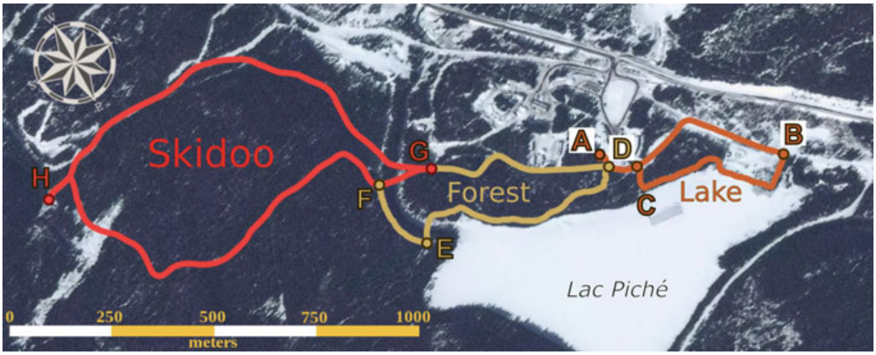


Fig. 3 The three large loops completed in the Montmorency forest in order to collect the `lake` (0.8 km), `forest` (1.3 km), and `skidoo` (2 km) datasets. The map was adapted from ©Mapbox and ©OpenStreetMap contributors

each time branches came near the sensor. Similarly, the dataset `forest` consists of a pedestrian trail and a cross-country skiing trail. In the first part of the trajectory, a pedestrian trail in the dense forest was traversed with a sleigh (`D` to `E`), followed by an untapped path through an even denser forest (`E` to `F`), and finally the sleigh was attached to a snowmobile and driven back to the starting point (`F` to `D`). Lastly, the dataset `skidoo` follows a 2 km long snowmobile trail. The data were gathered during a light snowfall. This loop was the easiest to map because it provided clear GNSS signal and was relatively flat from the beginning to the end (`G` to `H` to `G`).

5 Field Results

5.1 *Effects of Adding GNSS Penalty to ICP*

In order to inject the GNSS positioning information as a constraint into the ICP algorithm, it is necessary to find the correct transformation between the ICP map coordinate frame and the local tangent East-North-Up (ENU) frame. The translation, roll, and pitch angles are directly observable from the GNSS receiver and the IMU. The yaw angle (heading) measurement provided by the magnetometer part of the IMU is, however, affected by soft- and hard-iron errors induced by the mobile platform itself and by local deviations of the Earth's magnetic field. A practical solution to this problem was to observe a short initial portion of the GNSS trajectory to estimate the magnetometer heading offset. This way, the typical error between 15° and 20° could be reduced to 3° or less, which led to a satisfactory initial alignment. Another approach would be attaching a second GNSS receiver antenna to the mobile platform and estimating the heading angle from the relative positions of the two antennas. Both approaches, however, require a precise RTK GNSS solution. The standard uncorrected GNSS operating under the tree canopy yields excessive error and cannot be used for this purpose.

We first only applied a single penalty point to the ICP, based on the GNSS positioning. As the green trajectory in Fig. 4 demonstrates, this approach does not provide satisfactory results. On a short straight trajectory, we see that the single-point penalty forces the ICP to follow the GNSS reference, however, the orientation estimate drifts leading to a malformed map. In Fig. 4, this effect manifests itself as a slow rise in the pitch angle.

To fully constrain the ICP and avoid both orientation and position drift, we increased the number of penalty points to three, still following (8). The additional two points lie on the gravity and on the heading vectors as follows: In the map frame, one point lies below the GNSS position in the direction of the z axis. The second point lies in the x-y plane, in the direction of the current heading as indicated by the IMU and GNSS. In the scan frame, these two new points are accordingly projected using the IMU orientation information. In the ideal case, all three penalty points in the scan frame coincide with their map counterparts. Otherwise, the penalty is forcing the ICP solution toward the ideal state . The effect is demonstrated in Fig. 4 by the orange trajectory; the ICP output follows the GNSS positioning while keeping the correct orientation as well.

5.2 *The* Forêt Montmorency Dataset Results and Discussion

For each dataset, three mapping configurations were evaluated: GNSS+IMU (i.e., prior), ICP with penalties (i.e., penalty), and ICP without penalty (i.e., baseline). When processing, the `ethz-icp-mapping` was used with $r_{\max} = 100$ m and $\varepsilon =$

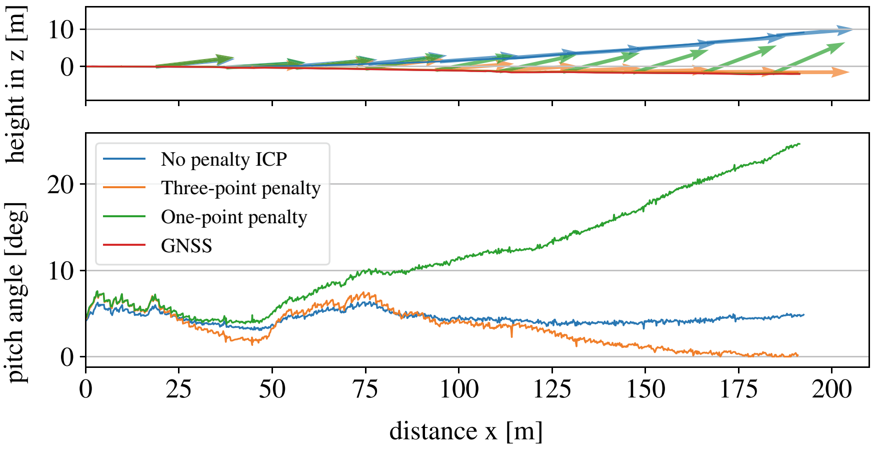


Fig. 4 Effect of adding penalty points to the ICP. *Top*: side view of a straight trajectory segment along with orientations represented by arrows. *Bottom*: corresponding pitch angles, unavailable for GNSS. The test was done in an open area with good GNSS coverage

5 cm for `lake` and `forest`. The `skidoo` dataset uses $\varepsilon = 10$ cm due to its size and memory requirements. The resulting maps are shown in Fig. 5.

`Lake`—In this dataset, the baseline map looks similar to the penalty map when observed from the top. However, a side view clearly shows that the buildings at the start and the end of the trajectory do not match. This effect is avoided by applying the penalty. For each configuration, we highlighted a pine tree in the top part of the map. While the tree is crisper in the baseline, the penalty map's tree is clearly sharper than the one from the prior map. **`Forest`**—The particularity of this map is the rough trail at the bottom of the map (see subfig. `III` and `IV` of Fig. 6). It suffers from large circular artifacts caused by the platform being immobile and by major changes of orientation along that trail. Again, the penalty is less clear than the baseline, but quite an improvement compared to the prior map. The circular-shaped building at the top part of the map is quite blurry in both the prior and penalty maps. This lack of crispiness is caused by the start and end of the trajectory not correctly matching. **`Skidoo`**—The baseline map is the most bent of all three datasets. This bending has two probable causes: (1) the map was created with lower density $\varepsilon = 10$ cm contrary to the other maps (2) the trajectory is twice as long as the `lake` dataset. Otherwise, the behavior is similar to that of the other datasets with the similar circular-shaped artifacts in the prior map.

None of the baseline maps manages to close the loop; they all drift and bend over time. Because of the magnetometer-based heading, the prior maps show large circular artifacts at several locations. These artifacts are especially noticeable in zones where the platform stops moving—the heading cannot be corrected by GNSS in this case. All penalty and prior maps closed the loop in a similar fashion. Moreover, the penalties manage to achieve a trade-off between global and local consistency.

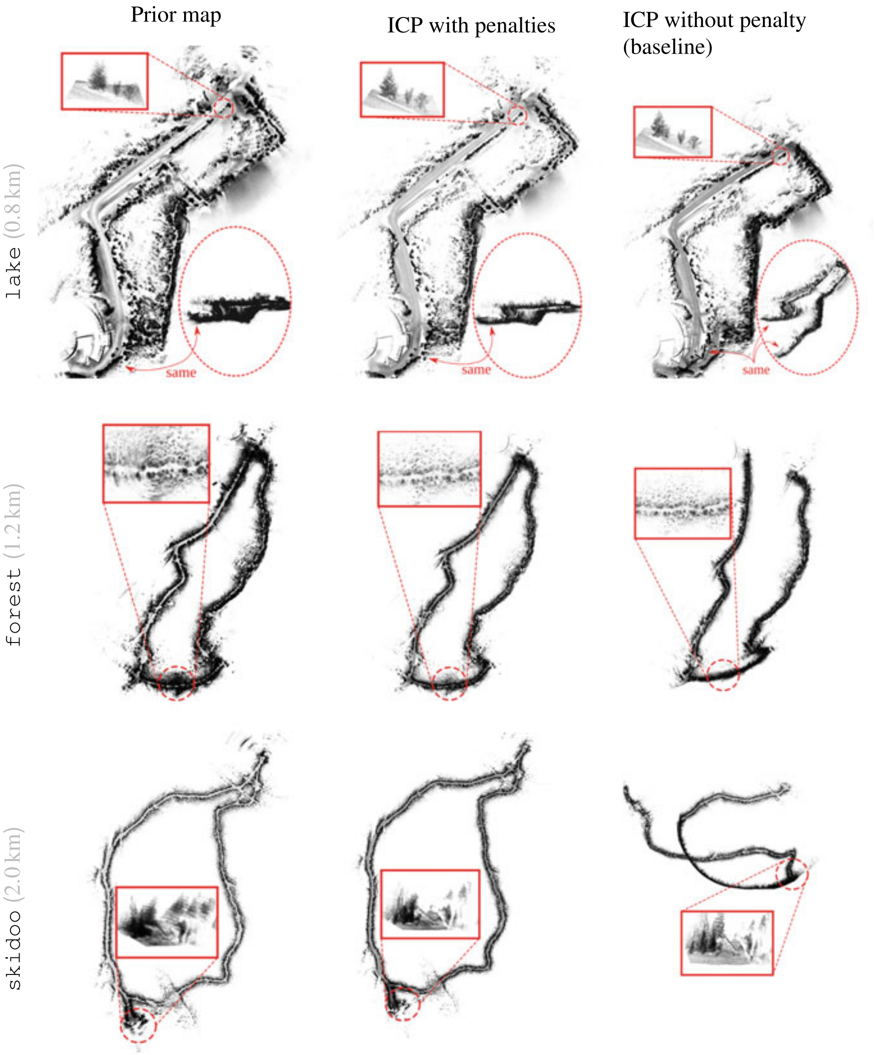

Prior map
ICP with penalties
ICP without penalty (baseline)
lake (0.8 km)
forest (1.2 km)
skidoo (2.0 km)
same
same
same

Fig. 5 Top view of the point cloud maps of the three datasets; prior, penalty and baseline configurations. For the `lake` dataset, a side view of the map (red dashed ellipse) shows the misalignment between the start and end. Red insets show the local (in)consistencies otherwise not visible at the full scale. Some trees in the insets are up to 15 m in height

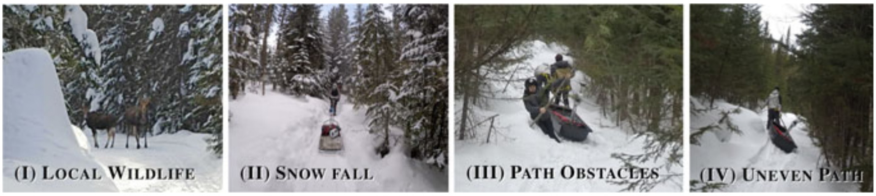


Fig. 6 Aspects of mapping a subarctic boreal forest. (`II`) shows snowfall caused by the snow in the tree branches. (`I`) was taken in `skidoo`, (`II`) in `lake`, and (`III`)–(`IV`) in `forest`

The field trials at the *Forêt Montmorency* presented a number of challenges. As `(I)` of Fig. 6 shows, local wildlife might hinder your experiments. We had a pair of moose blocking one of our trajectories and the experiment had to be rescheduled for another day. Also, wild birds used our static GNSS antenna as a perch. Another challenge is snowfall, as even a light snowfall will be visible on the map. Furthermore, even when our experiments were done on a clear day, they were still affected by snow falling from the trees (see subfig. `II` of Fig. 6). In the `forest` trajectory, we had to pass through an untapped trail with trees blocking our way `(III)`. Because of the roughness of that trail, the platform almost tumbled over multiple times, `(IV)`. The snowmobile trails, on the other hand, were easy to pass through.

Cold can cause hindering issues with lidars that are not properly rated for low temperatures. We tested a lidar rated for a minimum of −10 °C, with exterior temperature during our field experiments varying between −17 °C and −7 °C. The lidar started to malfunction when the temperature dropped below 0 °C, producing spurious measurements to a level where the environment could not be seen anymore. Providing an exterior source of heat could mitigate the problem temporarily.A second lidar rated for −20 °C was used for our final experiments. Through the development process, we observed that sun glare was more apparent at low temperatures. More tests are required to fully understand the impact of cold on lidar, but one should be careful regarding the temperature ranting of sensors deployed as it can cause serious safety issues.

5.3 Steps Toward Real-Time Large-Scale Lidar Mapping

The large-scale point cloud maps bear several problems that complicate real-time deployment on autonomous vehicles . The processing time required to register the scans and update the map may limit the agility of the vehicle. Moreover, memory management needs to be taken into consideration. For example, the $\sim$23700000 data points of the `forest` final prior map consumed 1.8 GB when stored in RAM. To improve the mapping speed, we have implemented the $r_{\max}$ cut-map radius as defined in Sect. 3.3. As shown in Fig. 7, the mapper execution time is reduced. One can observe a flattening in the number of points in the cut-map around 250 to 500 s. This situation occurs when the mobile platform is immobile.

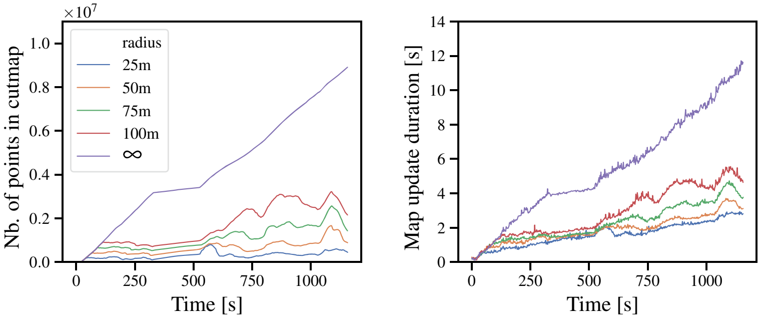


Fig. 7 Effect of the cut-map radius $r_{\max}$ on the complexity over time for the `lake` trajectory. *Left*: number of points used as a reference point cloud for the ICP. *Right*: computation time used by the mapper for each new lidar scan

Finally, in order to achieve globally consistent maps, we used the RTK GNSS solution consisting of two receivers, one static and the other attached to the mobile platform. The precise positioning information is obtained by combining the information from both receivers. In our case, it was done during the post-processing of the dataset. For real-time deployment, it is necessary to reliably transmit the static receiver information to the mobile vehicle, which may be difficult in dense vegetation.

6 Conclusion

In this paper, we explored the process of creating large globally consistent maps of subarctic boreal forests by adding external constraints to the ICP algorithm. The maps remained crisp even through 4.1 km of narrow walkable and snowmobile trails. We also discussed problems encountered with the environment and the lidar sensor during the field trials. Moreover, we introduced a computation optimization for very large maps, allowing real-time deployments. Encouraged by the results, this opens the door to further comparison with Normal Distribution Transformation (NDT) and GICP using better experimental validation and external tracking systems. Furthermore, studying the impact of penalties within ICP against graph minimization would lead to a better understanding of their pros and cons.

Acknowledgements This work was financed by Fonds de Recherche du Québec – Nature et technologies (FRQNT) and Natural Sciences and Engineering Research Council of Canada (NSERC) through the grant CRDPJ 527642-18 SNOW.

References

1. Zlot, R., Bosse, M.: Efficient large-scale three-dimensional mobile mapping for underground mines. JFR **31**(5), 758–779 (2014)
2. Pierzchała, M., Giguère, P., Astrup, R.: Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM. Comput. Electron. Agri. **145**, 217–225 (2018)
3. Williams, S., Parker, L.T., Howard, A.M.: Terrain reconstruction of glacial surfaces: robotic surveying techniques. RA Magaz. **19**(4), 59–71 (2012)
4. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. TPAMI **14**(2), 239–256 (1992)
5. Pomerleau, F., Colas, F., Siegwart, R., Magnenat, S.: Comparing ICP variants on real-world data sets: open-source library and experimental protocol. Auton. Robot. **34**(3), 133–148 (2013)
6. Thrun, S., Montemerlo, M.: The graphSLAM algorithm with applications to large-scale mapping of urban structures. IJRR **25**(5–6), 403–429 (2006)
7. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., et al.: Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. T-RO **32**(6), 1309–1332 (2016)
8. Landry, D., Pomerleau, F., Giguère, P.: CELLO-3D: estimating the covariance of ICP in the real world. In: ICRA (2019)
9. Lever, J.H., Delaney, A.J., Ray, L.E., Trautmann, E., Barna, L.A., Burzynski, A.M.: Autonomous GPR surveys using the Polar Rover Yeti. JFR **30**(2), 194–215 (2013)
10. Jagbrant, G., Underwood, J.P., Nieto, J., Sukkarieh, S.: LiDAR based tree and platform localisation in almond orchards. In: FSR, pp. 469–483 (2015)
11. Williams, S., Howard, A.M.: Developing monocular visual pose estimation for arctic environments. JFR **71**(5), 486–494 (2009)
12. Paton, M., Pomerleau, F., Barfoot, T.D.: In the dead of winter: challenging vision-based path following in extreme conditions. STAR **113**, 563–576 (2016)
13. McDaniel, M.W., Takayuki, N., Brooks, C.A., Salesses, P., Lagnemma, K.: Terrain classification and identification of tree stems using ground-based LiDAR. JFR **29**, 891–910 (2012)
14. Wallace, L., Lucieer, A., Malenovsk, Z., Turner, D., et al.: Assessment of forest structure using two UAV techniques: a comparison of airborne laser scanning and structure from motion (SfM) point clouds. Forests **7**(3) (2016)
15. Tian, J., Reinartz, P., d'Angelo, P., Ehlers, M.: Region-based automatic building and forest change detection on cartosat-1 stereo imagery. ISPRS **79**, 226–239 (2013)
16. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. Int. J, Robot. Res. (2013)
17. Pomerleau, F., Colas, F., Siegwart, R.: A review of point cloud registration algorithms for mobile robotics. Found. Trends Rob. **4**(1), 1–104 (2015)
18. Ohta, N., Kanatani, K.: Optimal estimation of three-dimensional rotation and reliability evaluation. In: ECCV, pp. 175–187 (1998)
19. Estépar, R.S.J., Brun, A., Westin, C.-F.: Robust generalized total least squares iterative closest point registration. In: MICCAI (2004)
20. Maier-hein, L., Franz, A.M., Santos, T.R., Schmidt, M., Fangerau, M., et al.: Convergent ICP algorithm to accomodate anisotropic and inhomogenous localization error. TPAMI **34**(8), 1520–1532 (2012)
21. Billings, S.D., Boctor, E.M., Taylor, R.H.: Iterative most-likely point registration (IMLP): a robust algorithm for computing optimal shape alignment. PLoS ONE, 1–45 (2015)
22. Segal, A.V., Haehnel, D., Thrun, S.: Generalized-ICP. In: RSS (2009)
23. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: ICRA, pp. 2724–2729 (1992)
24. Babin, P., Giguère, P., Pomerleau, F.: Analysis of robust functions for registration algorithms. In: ICRA (2019)
25. Balachandran, R., Fitzpatrick, J.M.: Iterative solution for rigid-body point-based registration with anisotropic weighting. SPIE **7261**, 1–10 (2009)

26. Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R.: Long-term 3D map maintenance in dynamic environments. In: ICRA, pp. 3712–3719 (2014)
27. Laconte, J., Deschênes, S.-P., Labussière, M., Pomerleau, F.: Lidar measurement bias estimation via return waveform modelling in a context of 3d mapping. In: ICRA (2019)

Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem

Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto

1 Introduction

MAVs such as the DJI M600 shown in Fig. 1 present ideal platforms for supporting demining efforts and other critical remote sensing tasks. These commercially available platforms are fast to deploy and can collect useful data that are often inaccessible to other modes of sensing [19]. For demining applications, this means that an MAV carrying a GPSAR system can be flown over a minefield to collect in situ, high precision measurements for locating buried landmines [22].

At present, the target coverage area is 1 ha per battery charge. Furthermore, since the MAV must fly at relatively low altitudes to emit enough energy into the ground, structures in the environment such as trees and buildings can force no-fly-zones (NFZs) across the space. These obstacles can be especially problematic since they are often omitted from maps and must be dealt with at deployment time. This results in a need for an autonomous coverage planning and execution solution that can rapidly replan trajectories in the field.

R. Bähnemann (✉) · N. Lawrance · J. J. Chung · M. Pantic · R. Siegwart · J. Nieto
Autonomous Systems Lab, ETH Zurich, Zürich, Switzerland
e-mail: brik@ethz.ch

N. Lawrance
e-mail: lawrancn@ethz.ch

J. J. Chung
e-mail: chungj@ethz.ch

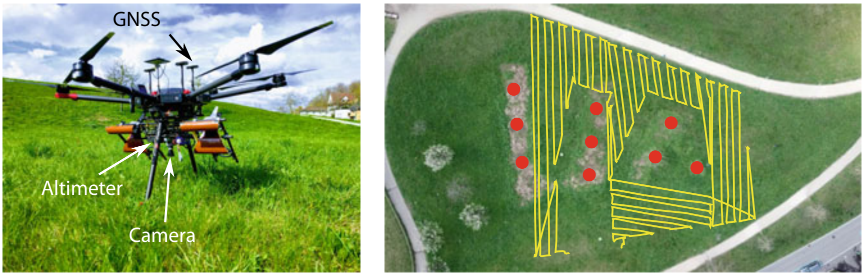
M. Pantic
e-mail: mpantic@ethz.ch

R. Siegwart
e-mail: rsiegwart@ethz.ch

J. Nieto
e-mail: nietoj@ethz.ch

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_20



(a) The MAV platform. (b) Coverage path over sloped terrain (trees in red).

Fig. 1 Deployment of our landmine detecting MAV. The low-altitude coverage path was generated in the field. The operator defined obstacle boundaries to avoid trees and the planner found an optimal path. Video: https://youtu.be/u1UOqdJoK9s

Many existing commercial mission planners, e.g., ArduPilot Mission Planner,[1] Pix4DCapture,[2] Drone Harmony,[3] and DJIFlightPlanner,[4] provide implementations of 2D coverage planners which allow specifying polygon flight corridors and generating lawnmower patterns. However, these mission planners aim at high-altitude, open-space, top-down photography and thus neither allow specifying obstacles within the polygon nor consider the polygon edges as strict boundaries. Furthermore, due to the safety limitations of commercial platforms, most commercial planners enforce a minimum altitude which is above the required 3 m of our GPSAR system.

While solutions for generating coverage paths in general polygon environments exist, the implementations are either not publicly available, computationally unsuitable for complex large environments, or lack the full system integration that would allow a user to freely designate the flight zone.

In this work, we develop a complete pipeline for rapidly generating 2D coverage plans that can account for NFZs. Our proposed solution takes as input a general polygon (potentially containing NFZs) and performs an exact cell decomposition over the region. Initial sweep patterns are computed for each cell from which feasible flight trajectories are computed. Finally, the Equality Generalized Traveling Salesman Problem (E-GTSP) across the complete cell adjacency graph is solved to minimize the total path time.

Results comparing our proposed coverage planner to exhaustively searching a coverage adjacency graph demonstrate an order of magnitude speedup in the computation time for each trajectory. Specifically, we show that our solution's computation time grows reasonably with increasing map complexity, while the exact solution grows unbounded. Furthermore, the flight path computed by our method is 14% shorter than those found by solving a regular traveling salesman problem (TSP) while providing the same level of coverage.

---

[1] http://ardupilot.org/planner/.

[2] https://www.pix4d.com/product/pix4dcapture.

[3] https://droneharmony.com.

[4] https://www.djiflightplanner.com/.

The contributions of this work are the following:

- An E-GTSP-based fast 2D coverage planning algorithm that allows specifying polygonal flight zones and obstacles.
- Benchmarks against existing solutions.
- An open-source robot operating system (ROS) implementation.[5]

We organize the remainder of the paper as follows. Section 2 presents related work. In Sect. 3, we present the background in computational geometry to generate the sweep permutations. In Sect. 4, we present our coverage planner formulation to solve for the optimal coverage pattern. Finally, we validate our method in Sect. 5 before we close with concluding remarks in Sect. 6.

2 Related Work

Coverage planning in partially known environments is an omnipresent problem in robotics and includes applications such as agriculture, photogrammetry, and search. It is closely related to the art gallery problem (AGP) and TSP, which are NP-hard [18, 20], and for which heuristic solutions have been developed to cope with high dimensional problems. The coverage planning problem can be stated as follows: given a specified region and sensor model, generate a plan for a mobile robot that provides complete coverage, respects motion and spatial constraints, and minimizes some cost metric (often path length or time). The underlying algorithms to solve sweep planning in a 2D polygon map are usually either based on approximate or exact cellular decomposition. The state of the art in robotic coverage planning is summarized in [3, 9].

Early work in robot coverage planning developed approximate cellular decomposition methods. In [27], the target region is first decomposed into connected uniform grid cells such that coverage is achieved by visiting all the cells. The authors presented a coverage wave front algorithm that obeyed the given starting and goal cells. Unfortunately, grid-based methods grow linearly in memory with the area to be covered and exponentially in finding an optimal path, making them unsuitable for large areas with small sensor footprints. Furthermore, guiding the wave front algorithm to pick a route that meets all mission requirements involves designing and tuning a specific cost function for each application. Spanning trees are an alternative method to solve this problem [7]. However, by default those result in a rather large number of turns which is undesirable for MAVs.

A simple approach stems from the idea that a polygonal region can be covered using sequential parallel back-and-forth motions, i.e., lawnmower or boustrophedon paths. This approach for 2D top-down lawnmower patterns in polygonal maps dominates commercial aerial remote sensing. The user defines a polygonal target region and a specified sweep direction, and the planner generates a path consisting of a sequence of equally spaced sweep lines connected by turns. The generated coverage paths are time-efficient, complete, and intuitive to the user. However, these planners cannot account for NFZs nor do they satisfy our targeted flight altitude requirements. Furthermore, except for ArduPilot Mission Planner, these commercial systems do not allow modification as they are closed source.

---

[5] https://github.com/ethz-asl/polygon_coverage_planning.

Exact cellular decomposition is a geometric approach that can handle NFZs by dividing a configuration space into simpler component cells, whose union is the complete free space. A full coverage path can then be found by creating a boustrophedon path in each cell and connecting all the cells. The individual cell coverage plans are connected in an adjacency graph, and solving the resulting TSP solves the coverage problem. Compared to grid-based methods, exact decomposition generally results in significantly fewer cells in simple large environments and thus a smaller TSP. Choset and Pignon [4] present the standard solution in 2D environments, which has been adapted by many planners for robot coverage [1, 8]. Several improvements have also been proposed to minimize the number of cell traversals and the number of turns along the coverage path. Namely, methods for optimizing over the sweep line direction rather than using a fixed direction for all cells have been proposed [16, 25]. However, these come at the cost of increased complexity when solving the optimal path through the cell-connectivity graph.

In our work, we revisit the exact cellular decomposition method and also generate multiple possible sweep directions per cell; however, we formulate the resulting search as an E-GTSP, allowing us to use a state-of-the-art genetic solver that handles significantly larger problem sizes. The E-GTSP, also known as TSP with neighborhoods, is a generalization of the classical TSP [17]. The goal is to find the shortest tour that visits exactly one node in each of a set of k neighborhoods. As first proposed by Waanders [26], we cluster all possible sweep patterns of a cell in a neighborhood and search for the shortest path that includes exactly one sweep pattern per cell.

Bochkarev and Lewis [2, 15] took this E-GTSP formulation even further. Instead of using per-cell predefined sweep patterns, they precompute a set of globally good sweeping directions for each monotone cell and build a graph over all individual straight segments where a neighborhood is defined by traversing a segment in either direction. Compared to our approach, their approach can give better solutions where the robot traverses from one cell to another cell without covering it completely first at the expense of a more complex E-GTSP and a predefined sweep direction.

To solve the E-GTSP, several options exist. Exact solvers like [6] only work reliably for small problem sizes. Converting the problem into a directed graph using a product graph [21] and solving it with an optimal graph solver, e.g., Dijkstra, basically falls back to solving it exhaustively as in [4, 25], which remains intractable for larger problem sizes. Thus, a practical solution for our problem sizes is to use heuristic solvers. Helsgaun [12] transforms the E-GTSP into a TSP and uses an approximate TSP solver. The Gutin and Karapetyan solver (GK), on the other hand, is a memetic algorithm that approximately solves the E-GTSP directly and is faster but with reduced performance for large problems [10].

3 Geometric Path Generation

In order to solve the coverage path problem, we follow the route of exact cellular decomposition techniques outlined in Fig. 2. We decompose a general polygon with holes (PWH) into cells, in a way that guarantees that each cell can be fully covered by simple boustrophedon paths. Our algorithm creates a permutation of sweeping directions for each cell and finds the shortest route that connects and covers every cell to define the *coverage path*.

3.1 Sweep Pattern Permutation

A *sweep pattern* describes the combination of *straight segments* and *transition segments* that covers a single polygon cell. A continuous parallel sweep pattern can be generated perpendicular to any monotone direction of a simple polygon. Without loss of generality, we can consider the monotone direction as the y-direction of the polygon. A polygon P is considered y-monotone, if any line in the x-direction intersects P at a single point, a segment or not at all. In other words the line intersects P at most twice [5].

Figure 3a shows the straight segment generation in a y-monotone polygon. We initialize the first straight segment at the bottommost vertex parallel to the x-axis, which we refer to as *sweep direction*. In general, we restrict the sweep directions to be collinear to one of the edges of our polygon, as these directions have been proven to lead to a minimum number of straight segments to cover the polygon [13].

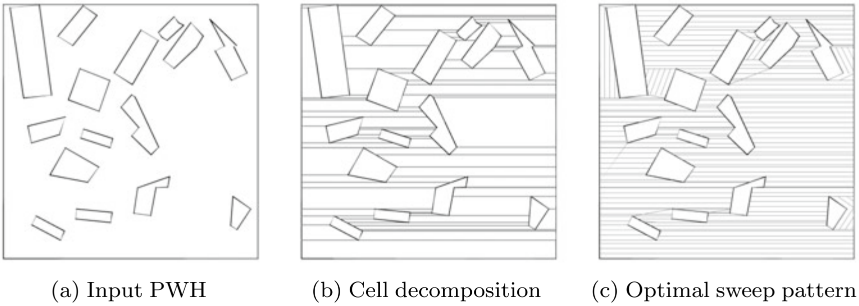
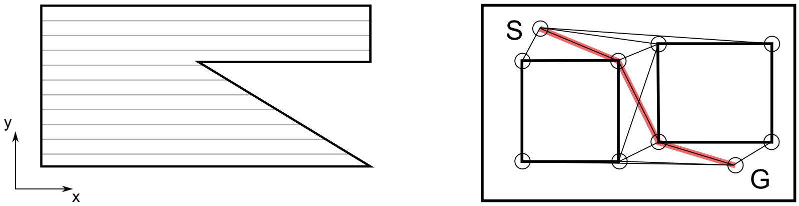

(a) Input PWH (b) Cell decomposition (c) Optimal sweep pattern

Fig. 2 The coverage algorithm on a synthetic map with 15 obstacles and 52 hole vertices



(a) Straight segments in a y-monotone polygon. (b) Shortest path transition segments.

Fig. 3 A sweep pattern consists of a set of parallel offset straight segments sequentially connected via transition segments. The straight segments are generated along edges that are perpendicular to a monotone direction (a). The transition segments are Euclidean shortest paths along the reduced visibility graph (b)

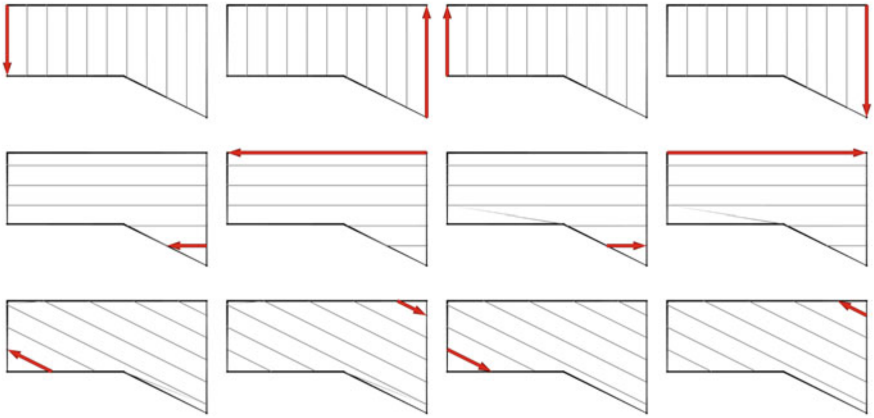
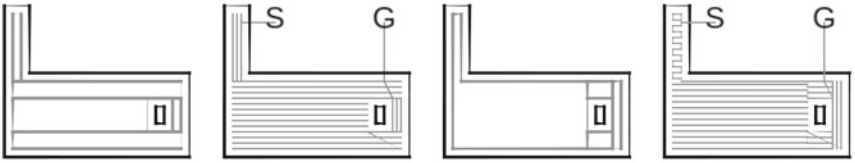


Fig. 4 Each *sweepable* direction (rows) has four *sweep permutations* (columns) based on the start vertex and start direction (red arrow)

The individual straight segments are generated by alternating between intersecting a line in the x-direction with the polygon and offsetting the line from the bottommost toward the topmost vertex. The sensor footprint hereby defines the offset distance.

Based on this construction criterion, the straight segments can start clockwise or counter-clockwise at the bottom vertex and go from the bottom to the top or from top to bottom. Thus, we generate four possible sweep patterns per *sweepable* direction as shown in Fig. 4.

To connect two straight segments (and later to connect two sweep patterns), we calculate the Euclidean shortest path that avoids collision with the NFZ. The Euclidean shortest path in a PWH is computed along the *reduced visibility graph* [14]. Figure 3b shows an example solution (red) using A* to search the graph [11]. The graph node set (circles) consists of all concave hull vertices and convex hole vertices.



(a) TCD (b) BCD

Fig. 5 Qualitative comparison of TCD and BCD. While both adjust well to rectangular environments, the BCD leads to fewer cells and redundant sweeps but can have degenerate sweeping behavior in narrow regions

3.2 *Polygon Decomposition*

To cover a general PWH, we partition it into monotone non-intersecting cells whose union is again the original PWH. In general, any monotone decomposition would be feasible but we only consider the trapezoidal cell decomposition (TCD) and the boustrophedon cell decomposition (BCD) [4, 5]. Both decompositions have different advantages as shown in Fig. 5. The TCD provides a partitioning that adjusts well in rectangular scenes with multiple directions of extension. The BCD also adjusts well to rectangular scenes and usually leads to fewer cells and thus fewer redundant sweeps and traversal segments [4]. On the downside, it can lead to degenerate sweeping behavior in cells with narrow protrusions.

Since both decompositions result from scan line algorithms, the scan line direction determines the set of cells. To find a good decomposition direction, we calculate the decomposition for every individual edge direction. A potentially good decomposition is the decomposition with the smallest altitude sum w, where altitude refers to the monotone extension of a cell. The minimum altitude sum corresponds to a minimum number of sweeps in the case of a fixed sweep direction [13].

$$w = \sum_{i=1}^{m} y_{\max,i} - y_{\min,i}, \tag{1}$$

where m is the number of monotone cells, $y_{\max,i}$ is the y-coordinate of the uppermost vertex, and $y_{\min,i}$ is the y-coordinate of the lowermost vertex in a y-monotone polygon cell.

4 Coverage Path Planning

After decomposing the input PWH into simple cells and generating a set of sweep patterns for each cell, the planner has to find the shortest sequence of sweep patterns

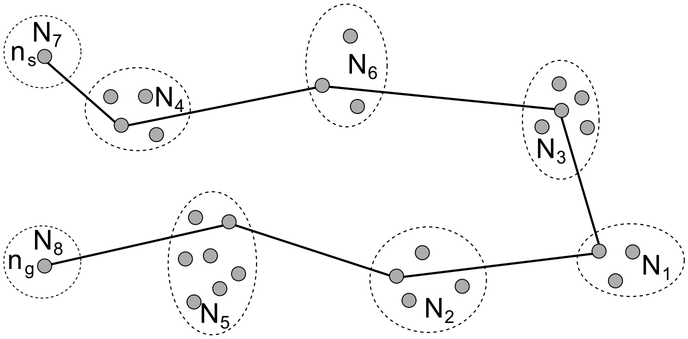


Fig. 6 Visualization of the coverage planning E-GTSP. The goal is to find the shortest path that visits exactly one sweep pattern (gray dot) in each polygon cell (dotted ellipsoid) while traveling from the start node n_s to the goal node n_g

such that every cell, and thus the whole PWH, is covered. To solve this problem efficiently, we formulate it as an Equality Generalized Traveling Salesman Problem.

Figure 6 sketches the adjacency graph $G = (N, A)$, where $N = \{n_1 \dots n_n\}$ is the node set and $A = \{(n_i, n_j) : n_i, n_j \in N, i \neq j\}$ is the set of directed arcs. The node set N is divided into m mutually exclusive and exhaustive clusters $N_1 \dots N_m$, i.e., $N = N_1 \cup \dots \cup N_m$ with $N_i \cap N_j = \emptyset \; \forall i, j \in \{1 \dots m\}, i \neq j$.

A node n_i represents an individual sweep pattern that covers a single monotone polygon cell as shown in Fig. 4 or the start or goal point. Every monotone cell (and the start and goal points) represents an individual cluster N_i. The arcs are the shortest path connecting the end of one sweep pattern with the start of the next sweep pattern in another cluster. The start node has outgoing arcs to all nodes, and the goal node has incoming arcs from all nodes.

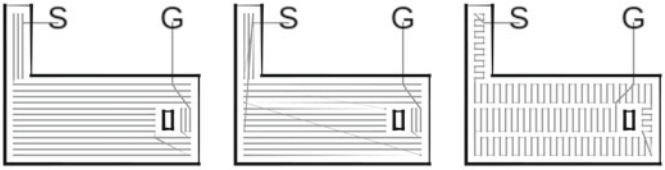
Every arc (n_i, n_j) has a non-negative cost c_{ij} which is the sum of the shortest path cost t_{ij} from n_i to n_j and the cost t_j to execute sweep pattern n_j.

$$c_{ij} = t_{ij} + t_j. \tag{2}$$

Because the start and the end of a sweep pattern do not coincide, the cost matrix $\boldsymbol{C} = (c_{ij})$ is asymmetric, i.e., $c_{ij} \neq c_{ji}$.

Since MAVs can hover and turn on the spot, we can plan rest-to-rest segments between waypoints which obey the straight-line assumption of our coverage planner. The trajectories are modeled with velocity ramp profiles with instantaneous acceleration and deceleration and a maximum velocity. The cost of a trajectory is the sum of all segment times. The segment time t between two waypoints is a function of the distance d, the maximum velocity v_{max}, and the maximum acceleration a_{max},

$$t = \begin{cases} \sqrt{\frac{4d}{a_{max}}}, & \text{for } d < 2d_a \\ 2t_a + \frac{d - 2d_a}{v_{max}}, & \text{for } d \geq 2d_a \end{cases}, \quad \text{where } t_a = \frac{v_{max}}{a_{max}}, \quad d_a = \frac{1}{2} v_{max} t_a. \tag{3}$$



(a) Time (b) Waypoints (c) Distance

Fig. 7 Qualitative comparison of different optimization criteria. Minimizing time results in elongated trajectories with short transition segments. Minimizing the number of segments also leads to long trajectories, but does not necessarily lead to good transitions. Minimizing only the Euclidean distance can lead to undesired sweeps since turns are not penalized

t_a is the time to accelerate to maximum velocity and d_a is the distance traveled while accelerating. Figure 7 shows that by tuning v_{max} and a_{max} with respect to our platform and sensor constraints, the optimization finds a good compromise between minimizing distance and minimizing the number of turns.

Algorithm 1 summarizes the process of setting up the E-GTSP problem. First, we decompose the polygon into monotone cells. For each cell, we compute the sweep permutations and make each sweep pattern a node in the graph where the neighborhood is defined by the cell ID. Finally, we create the edges between all sweeps patterns of different cells using the precomputed reduced visibility graph. Once the cost matrix is fully defined, we solve the E-GTSP using GK[6] as an off-the-shelf open-source solver [10].

4.1 Pruning

While our problem size (tens of clusters, hundreds of nodes, hundreds of thousands of edges) is no problem for GK, generating the edges is the bottleneck of the algorithm because every sweep pattern needs to be connected to almost any other sweep pattern through a Euclidean shortest path. The total number of arcs grows quadratically with the number of nodes.

Fortunately, the optimization problem is modular, i.e., any sweep pattern combination that visits every cell will achieve full coverage and the path cost is the only optimization criterion. We can safely prune a node $n_i \in N_k$ if it is cheaper to first traverse from n_i's start point to the start point of $n_j \in N_k$, perform coverage n_j, and return to n_i's goal point.

---

[6] https://csee.essex.ac.uk/staff/dkarap/?page=publications&key=Gutin2009a.

Algorithm 1 Generating the E-GTSP adjacency graph

```
Require: pwh, sweep_distance, wall_distance, cost_func, decomposition_type
Ensure: adj_graph
▷ Decompose polygon into monotone cells.
pwh = pwh.offsetPolygon(wall_distance)
cells = pwh.computeDecomposition(decomposition_type)
▷ Compute all sweep patterns per cell and create E-GTSP nodes.
vis_graph = pwh.computeVisiblityGraph()
for all cell ∈ cells do
    sweep_patterns = cell.computeSweepPatterns(sweep_distance, vis_graph)
    for all sweep_pattern ∈ sweep_patterns do
        n.sweep_pattern = sweep_pattern
        n.cell_id = cell.id
        n.start_vis = pwh.computeVisibility(sweep_pattern.start)
        n.goal_vis = pwh.computeVisibility(sweep_pattern.goal)
        adj_graph.addNode(n)
▷ Densely connect all nodes via Euclidean shortest paths.
adj_graph.prune(vis_graph)
for all from ∈ adj_graph.nodes do
    for all to ∈ adj_graph.nodes ∧ from.cell_id ≠ to.cell_id do
        e.path = vis_graph.solve(m.sweep.goal, m.goal_vis, n.sweep.goal, n.start_vis)
        e.cost = cost_func.compute(n.sweep_pattern) + cost_func.compute(e.path)
        adj_graph.addEdge(e)
```

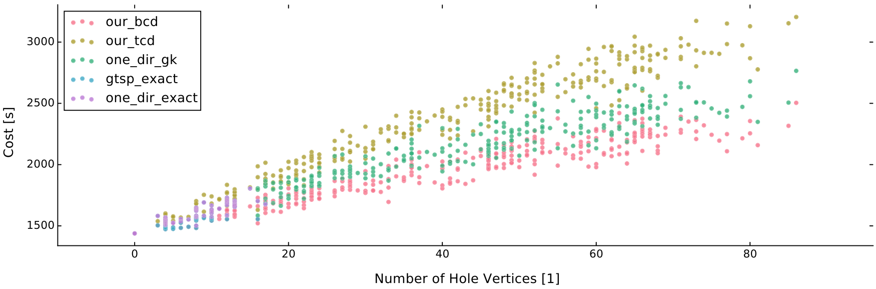
5 Results

The algorithm has been implemented in C++ using the Computational Geometry Algorithms Library (CGAL) [24]. CGAL provides efficient, reliable, and exact geometric algorithms which we extended to generate the sweep lines and the BCD. In order to find an exact solution, we implement [21] to convert the E-GTSP into a directed graph for an exhaustive exact solution using Dijkstra's algorithm.

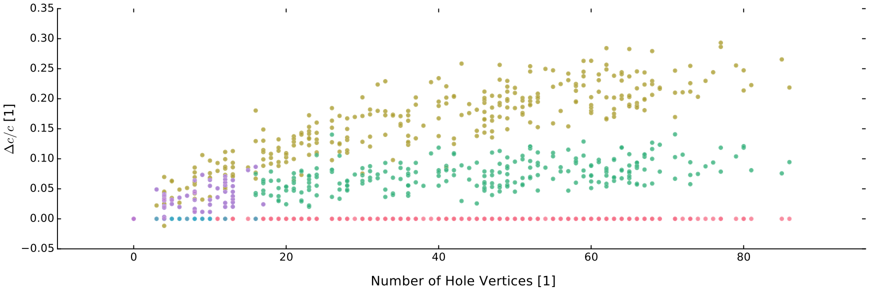
5.1 Simulation Benchmark

We set up a simulation benchmark to evaluate the performance of our algorithm (*our*) against the classical sweep planner with one sweep direction (*one_dir*) [4], to compare BCD and TCD, and to demonstrate the superiority of a designated E-GTSP solver over *exact* brute-force search. Our test instances, e.g., Fig. 2, are automatically generated from the EPFL aerial rooftop dataset [23] to provide polygon maps with realistic obstacles and dimensions. Our synthetic dataset consists of 320 rectangular worlds with an area of 1 ha and 0–15 rooftop obstacles. The benchmark was executed on an Intel® Core™ i7-7820HQ CPU @ 2.9 GHz.

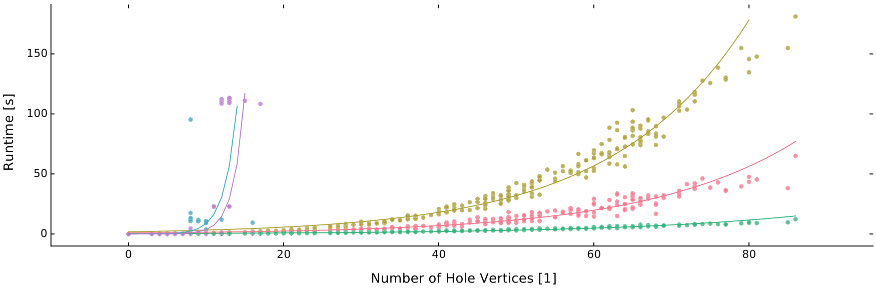
To relate our solution to the complexity of the maps, we plot coverage path cost and algorithm runtime against the number of hole vertices, as these are the events of



(a) The absolute path cost.



(b) The relative improvement of our planner.



(c) The computation times.

Fig. 8 Benchmark results for increasing map complexity. Our E-GTSP with BCD generally has the lowest path cost and reasonable computation times even for complex maps. Exact solutions only work for maps with a few holes. The TCD is generally worse than BCD

the decomposition scan line algorithms. Figure 8a shows the trajectory cost of the different planner configurations. Because our planner takes advantage of different sweep directions and start points, it gives better results than the classic solution with only one fixed sweep pattern per cell. Furthermore, BCD leads to shorter trajectories than TCD because it generates fewer cells and thus fewer redundant sweeps at the adjacent edges and transition segments between the cells. On a side note, whenever the exact solution is available it coincides with the approximate solution from GK.

Table 1 Detailed computation times for the instance in Fig. 2 with 15 holes (52 hole vertices). Creating all edges is the greatest computational effort. The reduced problem thus has the smallest computation time. Our planner results in the lowest path cost

	Graph elements			Computation time (s)						Path cost (s)
	Cells	Nodes	Edges	Cells	Sweeps	Nodes	Pruning	Edges	Solve	
our_bcd	**52**	254	63056	5.68028	1.100210	2.984410	2.392840	22.2329	1.195960	**2391.21**
our_tcd	87	512	258704	6.92008	1.107450	4.940300	3.679660	84.6797	6.555060	2879.15
one_dir_gk	**52**	**52**	**2652**	5.53508	0.089057	0.246038	0.026695	**0.8067**	0.513497	2664.98

The relative differences in path cost between our planner and the other configurations are shown in Fig. 8b. We observe up to 14% improvement over the optimal fixed direction and 29% improvement of BCD over TCD. Figure 8c shows the computation time of the planners. Exact solutions fail to solve within 200 s for any of our scenarios with more than 20 hole vertices. Generating the product graph from the adjacency graph G and Boolean lattice of possible cluster sequences grows exponentially in the number of clusters [21]. The reduced problem *one_dir_gk* achieves the best computation time. At most, it takes 12 s for the most complex map. The full E-GTSP with BCD solves it within 65 s which is still reasonable for employing the planner in field experiments. Table 1 reveals that generating the dense E-GTSP graph is the greatest computational burden.

5.2 Experiment

We validate our planner in a real flight on a DJI M600 Pro. Our drone covers a non-convex open area with sparse trees, depicted in Fig. 1, in a low-altitude mapping scenario. The flight corridor is selected in a georeferenced map; an optimal sweep path is calculated with our planner and executed under GNSS control. The drone follows a velocity ramp profile as described in Sect. 4 and additionally turns at every waypoint in the flight direction. The slope of the terrain is not considered during planning. To regulate the height above ground level (AGL), we fuse Lidar and Radar altimeter data into a consistent altitude estimate. Table 2 shows the general experimental setup. The sweep distance is chosen based on image overlap. The velocity and acceleration are chosen to meet controller constraints and avoid motion blur. To validate coverage with a nadir configuration sensor, we record QXGA top-down imagery and generate the digital terrain model (DTM) shown in Fig. 9 using the Pix4D mapping tool. The DTM shows good coverage of the designated area but imperfect reconstruction due to instantaneous acceleration movements. Collision-free trajectory smoothing may improve the turning maneuvers both in speed and smoothness and consequentially increase the overall performance of the coverage planner.

Table 2 Flight experiment parameters

Area	Flight time	AGL	Sweep offset	$v_{\max}$	$a_{\max}$
1950 m^2	1050 s	4 m	1.5 m	3.0 ms^{-1}	0.5 ms^{-2}

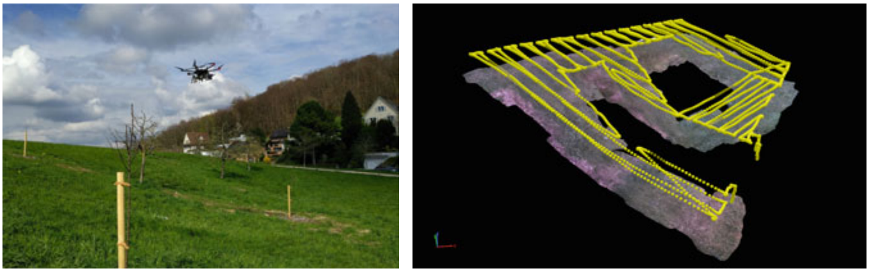

(a) Terrain following using altimeter data. (b) A Pix4D DTM reconstruction.

Fig. 9 The platform is capable of covering sloped terrain using the proposed planner

6 Conclusion

In this work, we presented a boustrophedon coverage path planner based on an E-GTSP formulation. We showed in comprehensive benchmarks on realistic synthetic polygon maps that our planner reliably solves complex coverage tasks in reasonable computation time, making it suitable for field deployment. Furthermore, we showed that our planner outperforms the classic boustrophedon coverage planner in terms of path cost. We validated in a field experiment the usability of our coverage algorithm on a real MAV and show that we can cover a 1950 m^2 area with obstacles at low altitude. Future work includes optimizing coverage for a side looking GPSAR configuration, collision-free trajectory smoothing to improve turning times, and integration into an airborne mine detection system.

Acknowledgements This work is part of the FindMine project and was supported by Urs Endress Foundation. The authors would like to thank their student Lucia Liu for her initial work on the BCD, Florian Braun and Michael Riner-Kuhn for their hardware support, and the reviewers for their constructive advice.

References

1. Bähnemann, R., Schindler, D., Kamel, M., Siegwart, R., Nieto, J.: A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects. In: IEEE International Symposium on Safety, Security and Rescue Robotics, pp. 123–128. IEEE (2017)
2. Bochkarev, S., Smith, S.L.: On minimizing turns in robot coverage path planning. In: IEEE International Conference on Automation Science and Engineering, pp. 1237–1242. IEEE (2016)

3. Cabreira, T., Brisolara, L., R Ferreira, P.: Survey on coverage path planning with unmanned aerial vehicles. Drones **3**(1), 1–38 (2019)
4. Choset, H., Pignon, P.: Coverage path planning: the boustrophedon cellular decomposition. In: Field and Service Robotics, pp. 203–209. Springer (1998)
5. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational geometry. In: Computational Geometry, pp. 1–17. Springer (1997)
6. Fischetti, M., Salazar González, J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper. Res. **45**(3), 378–394 (1997)
7. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math. Artif. Intell. **31**(1–4), 77–98 (2001)
8. Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., Ridao, P.: Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. J. Field Robot. **32**(7), 952–983 (2015)
9. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
10. Gutin, G., Karapetyan, D.: A memetic algorithm for the generalized traveling salesman problem. Nat. Comput. **9**(1), 47–60 (2010)
11. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
12. Helsgaun, K.: Solving the equality generalized traveling salesman problem using the Lin-Kernighan-Helsgaun algorithm. Math. Program. Comput. **7**(3), 269–287 (2015)
13. Huang, W.H.: Optimal line-sweep-based decompositions for coverage algorithms. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 27–32. IEEE (2001)
14. Latombe, J.C.: Robot Motion Planning. Springer Science & Business Media (1991)
15. Lewis, J.S., Edwards, W., Benson, K., Rekleitis, I., O'Kane, J.M.: Semi-boustrophedon coverage with a Dubins vehicle. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5630–5637. IEEE (2017)
16. Li, Y., Chen, H., Er, M.J., Wang, X.: Coverage path planning for UAVs based on enhanced exact cellular decomposition method. Mechatronics **21**(5), 876–885 (2011)
17. Mitchell, J.S.: Geometric shortest paths and network optimization. Handb. Comput. Geom. **334**, 633–702 (2000)
18. O'Rourke, J., Supowit, K.: Some NP-hard polygon decomposition problems. IEEE Trans. Inf. Theory **29**(2), 181–190 (1983)
19. Pajares, G.: Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs). Photogramm. Eng. Remote Sens. **81**(4), 281–329 (2015)
20. Papadimitriou, C.H.: The Euclidean travelling salesman problem is NP-complete. Theor. Comput. Sci. **4**(3), 237–244 (1977)
21. Rice, M.N., Tsotras, V.J.: Exact graph search algorithms for generalized traveling salesman path problems. In: International Symposium on Experimental Algorithms, pp. 344–355. Springer (2012)
22. Schartel, M., Burr, R., Mayer, W., Docci, N., Waldschmidt, C.: UAV-based ground penetrating synthetic aperture radar. In: IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, pp. 1–4. IEEE (2018)
23. Sun, X., Christoudias, C.M., Fua, P.: Free-shape polygonal object localization. In: European Conference on Computer Vision, pp. 317–332. Springer (2014)
24. The CGAL Project: CGAL User and Reference Manual, 4.13 edn. CGAL Editorial Board (2018). https://doc.cgal.org/4.13/Manual/packages.html
25. Torres, M., Pelta, D.A., Verdegay, J.L., Torres, J.C.: Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. Expert Syst. Appl. **55**, 441–451 (2016)
26. Waanders, M.: Coverage path planning for mobile cleaning robots. In: 15th Twente Student Conference on IT, pp. 1–10 (2011)
27. Zelinsky, A., Jarvis, R.A., Byrne, J., Yuta, S.: Planning paths of complete coverage of an unstructured environment by a mobile robot. Int. Conf. Adv. Robot. **13**, 533–538 (1993)

Fast Exploration Using Multirotors: Analysis, Planning, and Experimentation

Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael

1 Introduction

Fast and safe exploration of unknown environments is an important aspect of robotics applications such as urban search and rescue and disaster response. In such scenarios, it is essential for a robot or a team of robots to find survivors or objects in an unstructured and unknown environment quickly while maintaining collision-free operation. Aerial robots are particularly suitable for this task and have been used for mapping hazardous environments [16] and in prior work on exploration scenarios [1, 3, 6, 7]. Despite this interest, relatively little is known about the relationship between the dynamics of aerial robots and performance in exploration such as what limits performance in these systems or what can be done to improve performance in aerial exploration. This work investigates these limitations and to design planners suitable for exploration that satisfy these constraints.

Most works on robotic exploration study ground robots with largely trivial dynamics [11, 19, 20] and have been unconcerned with the role of speed in exploration. However, given the increasing application of aerial robots as sensing platforms, recent works have begun to consider the effects of system dynamics on high-speed

---

This work was supported by DOE (DE-EM0004067) and industry.

K. Goel (✉) · M. Corah · C. Boirum · N. Michael
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: kshitij@cmu.edu

M. Corah
e-mail: micahcorah@cmu.edu

C. Boirum
e-mail: cboirum@cmu.edu

N. Michael
e-mail: nmichael@cmu.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_21

exploration and navigation in unknown environments. Cieslewski et al. [6] propose a strategy based on maintaining rapid forward motion by driving the system toward frontier cells (cells on the boundary of free and unknown space [20]) within the camera field of view. While the authors note that reaction time for obstacle avoidance can limit speeds in exploration, they provide little discussion of why this happens or how it can be avoided. This work considers a broader variety of sensing actions that can avoid these limitations and also incorporates sensing and planning time into action design. Also, [12] provide a similar discussion as we do regarding how time delay in processing sensor data together with constrained dynamics can limit velocities when navigating unknown environments.

This work applies an information-based planning approach which reasons explicitly about information gain from future sensor observations [3, 4, 11]. As in previous works on robotic exploration [7, 8, 11], we use a randomized planning strategy based on Monte Carlo tree search [2, 5]. Here, we focus on the design of a library of motion primitives suitable for exploration at high speeds given the relationship between sensing constraints and maximum safe velocities. We evaluate this approach in simulation experiments demonstrating the exploration of a large warehouse environment and via outdoor field tests with a hexarotor robot.

2 Steady-State Velocity Analysis

This section presents an analysis of exploration performance for an aerial robot operating for steady-state conditions such as continuous motion toward a frontier. This analysis produces bounds on velocity and rates of entropy reduction, given constraints on dynamics and sensing. We leverage these insights in Sect. 3 to design motion primitive actions for rapid exploration.

2.1 System Model and Safety Constraints

This work applies a simplified double-integrator quadrotor model with acceleration and velocity constraints for analysis of limits on exploration performance, which can be thought of as a relaxation of dynamics models that are commonly used for position and attitude control of multirotor vehicles [13, 15]. Let $\mathbf{r} = [x, y, z]^{\mathrm{T}}$ be the position of the vehicle in an inertial world frame $\mathcal{W} = \{\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}}\}$, and let the body frame be $\mathcal{B} = \{\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}, \mathbf{z}_{\mathcal{B}}\}$. Assuming small roll and pitch angles, and given the yaw angle ψ, the system state is $\boldsymbol{\xi} = [\mathbf{r}^{\mathrm{T}}, \psi, \dot{\mathbf{r}}^{\mathrm{T}}, \dot{\psi}]^{\mathrm{T}}$. The derivatives of position and yaw satisfy bounds on velocity and acceleration

$$||\dot{\mathbf{r}}||_2 \leq V_{\max} \qquad ||\ddot{\mathbf{r}}||_2 \leq A_{\max} \qquad |\dot{\psi}| \leq \Omega_{\max} \tag{1}$$

where $||\cdot||_2$ is the L2-norm.

Further, the robot is equipped with a forward-facing depth sensor with a range of r_{depth} for use in mapping. However, while navigating the environment, the robot must also be able to avoid collisions with obstacles.

The requirement for collision-free operation restricts the set of actions that a multirotor can safely execute while navigating in an unknown environment. A planning policy can ensure collision-free operation by guaranteeing that the robot is able to stop entirely within unoccupied space $\mathcal{X}_{\mathrm{free}}$, given an appropriate collision radius r_{coll}, such as in the work of [9]. In the worst case, any voxel in the unknown space $\mathcal{X}_{\mathrm{unk}}$ may be revealed to be occupied and so possibly force the robot to stop within $\mathcal{X}_{\mathrm{free}}$.

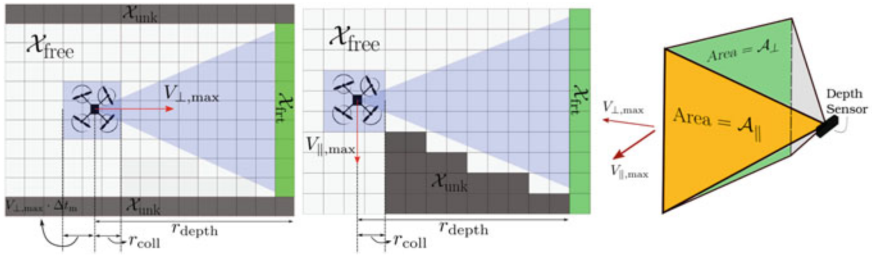
The robot plans once every Δt_{p} seconds, and there is also latency Δt_{m} for acquiring depth information and integrating it into the occupancy map. The sensor data is Δt_{m} seconds old at the beginning of planning, and once the planner is done, the robot executes the selected action for another Δt_{p} so that the total effective latency is no more than $\Delta t_{\mathrm{l}} = \Delta t_{\mathrm{m}} + 2\Delta t_{\mathrm{p}}$. Note that, although latency may be unpredictable in practice, the robot will not depend on consistent latency to maintain safe operation.

2.2 Steady-State Exploration Scenarios

Figure 1 illustrates two possible scenarios for steady-state motion with respect to a frontier. For the perpendicular case (Fig. 1a), the robot moves continuously toward a frontier and may have to avoid obstacles at the edge of the sensor range. As discussed in Sect. 2.1, the robot must be able to avoid collisions with obstacles in the unknown environment even if there are not any. This means that the robot must always be prepared to stop before reaching the frontier. For the parallel case (Fig. 1b), the robot moves along the frontier through space that has already been mapped. When known space is free of obstacles, the robot may continue to do so at the maximum allowable velocity. This scenario can also be thought of as the limit for a spiral motion—which will arise later in the experimental results—as the curvature becomes very small.

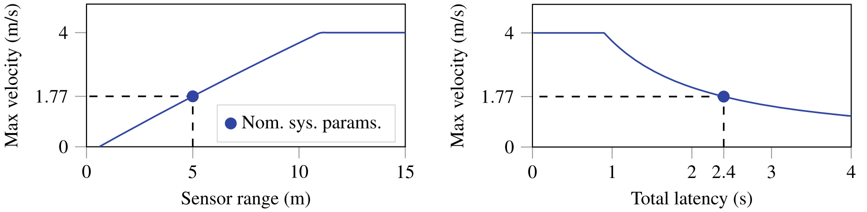
2.3 Bounds on Velocity

Given the system model and constraints for exploration scenarios, we now proceed with the calculation of the velocity bounds for motion perpendicular and parallel to the frontier, $V_{\perp,\mathrm{max}}$ and $V_{\parallel,\mathrm{max}}$, respectively. Maximum velocity toward the frontier is computed based on the motion at a constant velocity followed by stopping at maximum deceleration to satisfy the safety constraint. The expression for $V_{\perp,\mathrm{max}}$ is a function of acceleration (A_{max}), maximum sensing range (r_{depth}), the collision radius (r_{coll}), and the latency in planning Δt_{l} (see Fig. 1a) and is given by


$\mathcal{X}_{\text{unk}}$
$\mathcal{X}_{\text{free}}$
$V_{\perp,\max}$
$\mathcal{X}_{\text{frt}}$
$V_{\perp,\max} \cdot \Delta t_{\text{m}}$
$\mathcal{X}_{\text{unk}}$
r_{depth}
r_{coll}
$\mathcal{X}_{\text{free}}$
$V_{\parallel,\max}$
$\mathcal{X}_{\text{frt}}$
$\mathcal{X}_{\text{unk}}$
r_{depth}
r_{coll}
$V_{\perp,\max}$
$V_{\parallel,\max}$
Area = $A_\perp$
Area = $A_\parallel$
Depth Sensor

(a) Perpendicular scenario **(b)** Parallel scenario **(c)** Sensor cone

Fig. 1 Steady-state exploration scenarios. Analysis in Sect. 2 presents upper bounds on velocities are for a double integrator system **(a)** for motion perpendicular to a frontier ($V_{\perp,\max}$) and **(b)** for motion parallel to it ($V_{\parallel,\max}$). To ensure safety in the perpendicular case, the robot has to stop within a user-specified collision radius from the frontier ($\mathcal{X}_{\text{frt}}$), i.e. within $r_{\text{depth}} - r_{\text{coll}}$ from the current state. For the parallel case, no such restrictions exist since the robot is moving in the explored space which, ideally, is free ($\mathcal{X}_{\text{free}}$). **(c)** Combining the area of the projection of the sensor cone in the direction of motion with the bounds on velocity leads to upper bounds on rates of novel voxels observed during exploration



(a) Velocity ($V_{\perp,\max}$) vs. sensor range (r_{depth}) **(b)** Velocity ($V_{\perp,\max}$) vs. total latency (Δt_{l})

Fig. 2 Maximum achievable velocity moving toward a frontier ($V_{\perp,\max}$) according to (2) based on parameters used in Table 1 and varying **(a)** sensor range and **(b)** total latency (which consists of latency for the mapping system and time for planning). The circle marks the maximum velocity at which the robot can move toward unknown space for the parameters used in this paper (see Table 1) which is less than half the overall velocity bound. Approaching this velocity limit requires either sensor range exceeding 10 m, both decreased planning time and mapping latency, or some combination of the two

$$\begin{aligned} V_{\perp,\max} &= \min(V_{\max}, V'_{\perp,\max}) \\ V'_{\perp,\max} &= A_{\max} \cdot \left(\sqrt{\Delta t_{\text{l}}^2 + 2\frac{r_{\text{depth}} - r_{\text{coll}}}{A_{\max}}} - \Delta t_{\text{l}} \right). \end{aligned} \tag{2}$$

Figure 2 shows the variation of this bound with r_{depth} and Δt_{l} for the parameters used in this work. For motion parallel to a frontier (see Sect. 2.2 and Fig. 1b), there are no obstacles in the direction of motion. Therefore, the steady-state upper bound on the velocity moving parallel to the frontier is identical to the maximum achievable by the system, i.e. $V_{\parallel,\max} = V_{\max}$.

Table 1 Steady-state upper bounds on velocity and rate of entropy reduction for the scenarios described in Sect. 2.2. All values are computed for a planning time of 1 Hz, mapping latency 0.4 s, sensor point cloud of size 9.93 m × 5.68 m based on the RealSense D435 depth sensor with image size 424 px × 240 px, and a maximum depth r_{depth} of 5 m. Occupancy grid resolution is 20 cm with an overall bound on top speed $V_{\max}$ at 4 m/s, and collision radius r_{coll} set at 0.6 m

Value/Cases	Area (m^2)	Velocity (m/s)	Volume rate (m^3/s)	Entropy rate (bits/s)
Perpendicular ($\perp$)	56.4	1.77	99.83	1.25×10^4
Parallel ($\|$)	57.19	4.00	228.8	2.86×10^4
$\perp$, rapid yaw	56.8	1.77	100.5	1.26×10^4
$\|$, rapid yaw	78.05	4.00	312.2	3.90×10^4

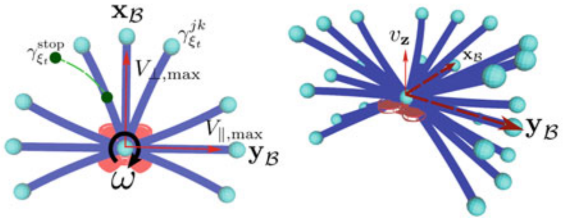
The entropy reduction then can also be bounded for each scenario in terms of the sensor geometry (see Fig. 1c) and steady-state velocities by projecting the sensing volume in the direction of motion. Here, we also introduce the possibility of rapid yaw motion during either motion. Results are shown in Table 1. Note that moving parallel to the frontier can provide significantly improved performance.

3 Action Representation

This section details the design of available actions for the proposed motion planning framework. We define a trajectory generation scheme, related parameters and conventions, and action design specifics leveraging insights gained in Sect. 2.

3.1 Motion Primitive Library Generation

Safe and accurate high-speed flight requires large and smooth linear acceleration and angular velocity references. Smoothness for such references depends on higher derivatives of position, jerk, and snap [14]. For this work, the actions that are available to the robot are snap-continuous, forward arc [21] motion primitives, which have previously been applied to high-speed teleoperation of multirotors [18]. Given the differentially flat state of the multirotor at time t, $\boldsymbol{\xi}_t = [x, y, z, \psi]^{\mathrm{T}}$ denote an available action parameterization as $\mathbf{a} = [v_{\mathbf{x}}, v_{\mathbf{z}}, \omega]$ where $v_{\mathbf{x}}$ and $v_{\mathbf{z}}$ are velocities in the body frame $\mathbf{x}_{\mathcal{B}}$ and $\mathbf{z}_{\mathcal{B}}$ directions, and ω is the body yaw rate. Actions are discretized using user-specified maximum velocity bounds in $\mathbf{x}_{\mathcal{B}} - \mathbf{y}_{\mathcal{B}}$ plane (ω variation, N_{ω} primitives) and in $\mathbf{z}_{\mathcal{B}}$ direction ($v_{\mathbf{z}}$ variation, $N_{\mathbf{z}}$ primitives) to obtain a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$ given by (Fig. 3)



(a) Variation in ω **(b)** Variation in $v_\mathbf{z}$

Fig. 3 Actions in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane at the multirotor state $\boldsymbol{\xi}_t$, $\gamma^{jk}_{\boldsymbol{\xi}_t}$ (blue) are generated using discretized velocity bounds obtained from the analysis in Sect. 2. The set of such primitives at each state is termed a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$. MPLs are sampled in directions perpendicular ($\mathbf{x}_\mathcal{B}$) and parallel ($\mathbf{y}_\mathcal{B}$, $-\mathbf{y}_\mathcal{B}$) to the sensor scans with speeds bounded by $V_{\perp,\max}$ and $V_{\parallel,\max}$, respectively; see (**a**). Variation in $\mathbf{z}_\mathcal{B}$ direction using a user-specified bound on vertical velocity, $V_\mathbf{z}$, yields the final action space shown in (**b**). Dynamically feasible stopping trajectories $\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$ are available for each primitive (green) for safety (only one shown for brevity)

$$\Gamma_{\boldsymbol{\xi}_t} = \{\gamma^{jk}_{\boldsymbol{\xi}_t} \mid j \in [1, N_\omega], k \in [1, N_\mathbf{z}], \|[v_\mathbf{x}, v_\mathbf{y}]\| \leq V_{\max}, \|v_\mathbf{z}\| \leq V_\mathbf{z}, \|\omega\| \leq \Omega_{\max}\} \tag{3}$$

where, $\|\cdot\|$ denotes L2-norm of a vector, $V_{\max}$ and $V_\mathbf{z}$ are the bounds on speed in $\mathbf{x}_\mathcal{B} - \mathbf{y}_\mathcal{B}$ plane and $\mathbf{z}$-direction, respectively, and $\Omega_{\max}$ is the bound on body yaw rate. For a given action discretization, the motion primitive $\gamma^{jk}_{\boldsymbol{\xi}_t}$ is an 8th order polynomial in time with fixed start and end point velocities and an unconstrained position at the end. The velocity at the end point, at time τ, is obtained by forward propagating unicycle kinematics using the current state and the action parameterization while higher order derivatives up to snap, end points are kept zero:

$$\dot{\boldsymbol{\xi}}_\tau = [v_\mathbf{x}\cos\psi, v_\mathbf{x}\sin\psi, v_\mathbf{z}, \omega]^\mathrm{T}, \;\; \psi = \omega\tau, \;\; \boldsymbol{\xi}^{(j)}_\tau = \mathbf{0}, \;\text{for } j \in \{2, 3, 4\} \tag{4}$$

where $\boldsymbol{\xi}^{(j)}$ denotes the jth time derivative of $\boldsymbol{\xi}$. The stopping trajectories at any $\boldsymbol{\xi}_t$ ($\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$, Fig. 3) can be sampled by keeping $\dot{\boldsymbol{\xi}}_t = \mathbf{0}$.

In contrast to the fixed duration (τ) primitives presented in [18], the duration for the primitive is kept minimum via a line search from the minimum possible duration (Δt_p) to the user-specified maximum duration ($\tau_{\max}$). The search terminates when the first dynamically feasible motion primitive is obtained. This dynamic feasibility check is based on pre-specified empirically observed bounds on linear acceleration and linear jerk $L2$-norms. This search achieves having the trajectory in the desired end point velocity $\dot{\boldsymbol{\xi}}_t$ in the minimum time possible from the current state.

3.2 Action Space for Fast Exploration

The action space for the proposed planner is a collection of MPLs, defined by (3), generated using linear velocities based on bounds obtained in Sect. 2, $V_{\perp,\max}$ and $V_{\parallel,\max}$.

Table 2 Motion primitive libraries used to construct action space using Eq. 3 from Sect. 3 and the bounds obtained after applying analysis presented in Sect. 2. The vertical velocity bound ($V_{\mathbf{z}}$) and the speed bound in $\mathbf{x}_{\mathcal{B}} - \mathbf{y}_{\mathcal{B}}$ plane (V_{max}) are kept at 0.3 m/s and 4.0 m/s, respectively. The total number of primitives for a MPL is $N_{\text{prim}} = N_{\omega} \cdot N_{\mathbf{z}}$

(a) Large library

ID	Type	Max. speed	Dir.	N_{ω}	$N_{\mathbf{z}}$	N_{prim}
1	Yaw	0	ψ	1	1	1
2	$\perp$	$V_{\perp,\text{max}}$	$\mathbf{x}_{\mathcal{B}}$	9	5	45
3	$\perp$	V_{max}	$\mathbf{x}_{\mathcal{B}}$	9	5	45
4	$\parallel$	V_{max}	$\mathbf{y}_{\mathcal{B}}$	9	5	45
5	$\parallel$	V_{max}	$-\mathbf{y}_{\mathcal{B}}$	9	5	45
6	Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	5	5

(b) Minimal library

ID	Type	Max. speed	Dir.	N_{ω}	$N_{\mathbf{z}}$	N_{prim}
1	Yaw	0	ψ	1	1	1
2	$\perp$	$V_{\perp,\text{max}}$	$\mathbf{x}_{\mathcal{B}}$	3	3	9
3	$\perp$	V_{max}	$\mathbf{x}_{\mathcal{B}}$	3	3	9
4	$\parallel$	V_{max}	$\mathbf{y}_{\mathcal{B}}$	3	3	9
5	$\parallel$	V_{max}	$-\mathbf{y}_{\mathcal{B}}$	3	3	9
6	Z	$V_{\mathbf{z}}$	$\mathbf{z}$	1	3	3

The planner uses 6 MPLs to represent the action space, $\mathcal{A}_{\text{act}} = \{\Gamma^{i}_{\xi_t} \mid i \in [1, 6]\}$, and sets of upper bounds on linear velocities (Table 2) define each of these different MPLs. These MPLs include both high-speed actions for navigating the environment and actions that mimic steady-state conditions described in Sect. 2.2. Later, in Sect. 5, we highlight effects on exploration performance due to these components, especially the high-speed parallel and low- speed perpendicular MPLs.

4 Action Selection

We formulate the action selection problem as a finite-horizon optimization seeking to maximize cumulative information gain [4], and build upon previous work [7, 8, 11] on robotic exploration using Monte Carlo tree search (MCTS).

Most MCTS-based planners follow four steps: selection, expansion, simulation playout, and backpropagation of statistics [2, 5]. Such planners usually construct a search tree iteratively by random rollout from a previously unexpanded node selected based on upper-confidence bounds for trees (UCT) [2]. Prior works [7, 8] have applied MCTS in planning for exploration using multirotors using a UCT-based selection policy, information gain rewards, and random simulation playout over a finite horizon. We extend this approach by adding considerations for model constraints into the node expansion phase of MCTS.

4.1 Information-Theoretic Exploration Objectives

Following a similar approach as our prior work [8], the planner optimizes an objective with two components: a local information reward based on Cauchy–Schwarz quadratic mutual information (CSQMI) [4], and a global reward for a decrease in the shortest path distance to points in the state space that are expected to provide significant information gain [8]. For any candidate action, γ_{ξ_t}, we compute the local information gain $\mathcal{I}_\gamma$ over user-specified time intervals and treat the joint information gain as a reward for the MCTS planner. The distance reward serves to direct the robot toward unexplored and possibly distant regions of the environment once the local environment is exhausted of information causing the local information reward to decrease. Alternatively, designers might substitute competing routing and scheduling approaches [10] for the distance cost subject to trade-offs in computational cost and system design.

4.2 Safety at High Speeds

Given the action representation described in Sect. 3, we require the planner to ensure safety while expanding nodes. Specifically, the trajectory tracked by the controller should both respect constraints on the dynamics and remain in known free space for all time. To satisfy this condition, before sending any trajectory to the robot, we require knowledge of a trajectory that will bring the robot to a stop—and potentially keep it there—afterward. As such, the robot will avoid collision, even if the planner fails to provide any output. If ever planning fails, the known stopping trajectory is sent to the robot, and the robot will continue to replan from a future reference point.

5 Results

This section describes hardware and simulation results for the proposed exploration approach. The simulation results evaluate performance in a warehouse-like environment which serves as a representative example of a large-scale exploration task. The hardware results demonstrate exploration at high speeds using a hexarotor platform under various degrees of clutter. Unless otherwise noted, the configuration of the robot for simulation matches the hardware.

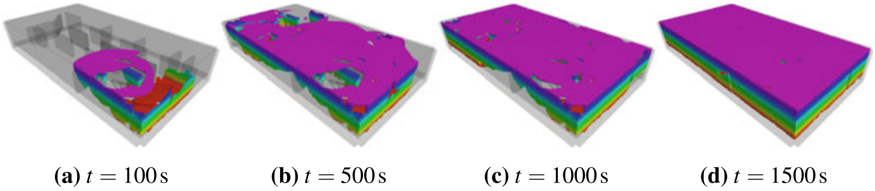
5.1 Aerial Platform

Platform used for experiments is a hexarotor aerial robot (Fig. 7a) with a forward-facing depth camera for mapping (RealSense D435) with a 89.57° by 59.24° field of view and a range of $r_{\text{depth}} = 5.0$ m. The robot itself weighs 55.37 N, has a thrust to weight ratio of 3.5, and has a diameter of 0.89 m from motor to motor. Limits on acceleration and jerk are set to $A_{\max} = 10\,\text{m/s}^2$ and $J_{\max} = 35\,\text{m/s}^3$, respectively, based on empirical data available for the platform. Unless otherwise stated, the planning horizon is kept at 4 s for all experiments. For both simulation and hardware experiments, mapping and planning run on a computationally constrained quad-core embedded CPU Gigabyte Brix 6500U. The robot obtains odometry estimates via a downward-facing camera and IMU using a monocular Visual-Inertial Navigation System (VINS-Mono [17]), previously used for high-speed teleoperation of a multirotor [18]. This state estimation component, only present for hardware experiments, is executed on a quad-core NVIDIA Tegra TX2 on-board the vehicle. Contrary to perfect state estimation in simulation experiments, for the hardware experiments the robot only has access to odometry for navigation and is susceptible to drift. For the purpose of this work, we will continue to emphasize the role of planning and speed in the exploration experiments and will comment briefly on ramifications of drift on outcomes. Future iterations of this platform will seek to combine a local mapping strategy [8] with a complete SLAM system.

5.2 Simulated Exploration of a Warehouse Environment

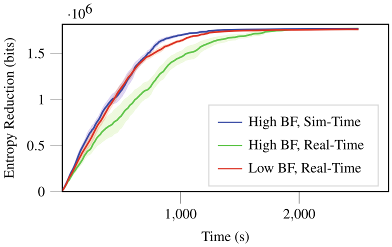
The simulations demonstrate the exploration of a large warehouse environment (pictured in Fig. 4). These trials are repeated for three system configurations which vary the motion primitive library and the computational setting:

- **High branching factor (BF), Sim-Time**: The planner uses the large motion primitive library (Table 2a) for exploration. The simulation and clock pause at the end of each planning round until the MCTS planner completes a user-defined number



(a) $t = 100\,\text{s}$ **(b)** $t = 500\,\text{s}$ **(c)** $t = 1000\,\text{s}$ **(d)** $t = 1500\,\text{s}$

Fig. 4 Occupied map at different stages of exploration of a simulated three-dimensional 60 m × 30 m × 11 m warehouse environment used for experiments. The map is colored based on Z height



Time to reach (s):	25%	50%	75%	90%
High BF, Sim-Time	**157.7**	**365.7**	**602.5**	**788.3**
High BF, Real-Time	267.4	501.6	866.5	1183.2
Low BF, Real-Time	193.2	382.5	604.8	901.1

Fig. 5 (Left) Entropy reduction versus time for each of the simulation trials. Transparent patches show standard error. (Right) Average time to reach fractions of the maximum entropy reduction over all trials (1.766×10^6 bits). While the High BF, Sim-Time case dominates in terms of entropy reduction, the Low BF, Real-Time case is able to provide similar performance. Note that the different configurations are described at the beginning of Sect. 5.2 and that BF denotes branching factor

(3000) of iterations. The simulation time then does not include this additional time spent in planning.

- **High BF, Real-Time**: The planner uses the large motion primitive library for exploration, but the simulation of the multirotor runs in real time. The planner runs in an anytime fashion on a computer comparable to the onboard computer used for flight experiments presented in Sect. 5.3 while simulators for the camera and dynamics run on a separate computer.
- **Low BF, Real-Time**: The planner uses the minimal motion library (Table 2b) for exploration and the computational configuration is the same as the above.

These experiments first establish baseline performance (High BF, Sim-Time) given access to a variety of motion primitives and a relatively large amount of planning time. The latter two configurations demonstrate online planning in a configuration that closely matches the hexarotor platform used in this paper. These experiments seek to demonstrate the role of computational constraints in the design of the motion primitive library. For each configuration, we provide several trials, one for each of 5 given starting locations.

Each trial lasts 2500 s which provides ample time to explore the entire environment. For all trials, the perpendicular velocity $V_{\perp,\max}$ is further limited to 1.25 m/s, below the value of 1.77 m/s obtained in Sect. 2 which we find admits forward motion at a constant speed given the trajectory generation approach used for motion primitive design. The maximum speed is set to more than three times greater at $V_{\max} = 4.0$ m/s.

Figure 5 summarizes exploration performance for each experiment. The high branching factor Sim-Time case which has access to extra planning time performs at least as well or better than the other configurations in terms of entropy reduction. However, the configuration with the same motion primitive library and real time is significantly impaired and requires between approximately 1.3–1.8 times as long to reach reported levels of entropy reduction. The lower branching factor case matches the first configuration much more closely. As such, this latter configuration is appropriate for deployment on the compute-constrained hexarotor platform.

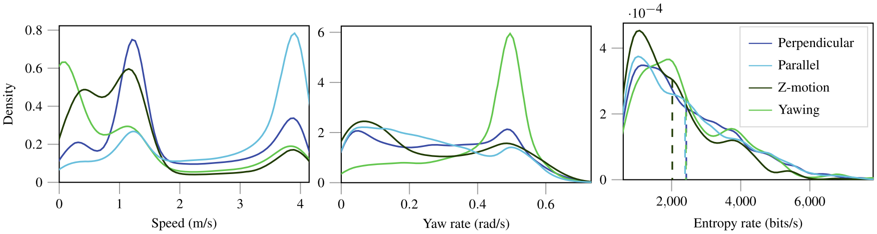


Fig. 6 Exploration performance by action. Plots provide estimates of probability densities (via kernel density estimation) for speed, yaw rate, and entropy rate conditional on the type of action (Table 2) being executed by the robot. All densities are also conditional on a significant entropy reduction rate (greater than 600 bits/s) in order to emphasize performance characteristics for actions that directly contribute to the map rather than traversal of known space or time periods after exploration is effectively complete. Note that, even though the option can have access to high-speed motions perpendicular to frontiers, entropy reduction for perpendicular actions occurs primarily at lower speeds (1.25 m/s) in accordance with the analysis in Sect. 2

In addition to being able to explore an environment rapidly and completely, we characterize the roles of the motion primitive actions in the exploration task. Figure 6 shows density estimate plots for speeds, yaw rates, and entropy rate labeled by the type of action selected by the MCTS planner for execution for periods when the entropy reduction rate is significant (greater than 600 bits/s) so as to separate exploration actions from the traversal of the environment and time periods after exploration is effectively complete. This threshold on the entropy reduction rate corresponds to a knee point in the overall distribution of entropy reduction rates: 94.4% of all entropy reduction occurs above this threshold but during only 27.9% of time during the trials. Interestingly, the time rate of entropy reduction is largely consistent across action types. However, as expected, motions perpendicular to the frontier primarily contribute to entropy reduction at reduced speed despite both low-speed and high-speed primitives being available. Table 3 shows further statistics for the different kinds of actions. Even though entropy reduction rates are similar across actions when the entropy reduction rate is significant, the planner selects motions parallel and perpendicular to frontiers most frequently, and those actions account for more than 80% of all entropy reduction.

5.3 Hardware Experiments Under Varied Conditions

Real-world autonomous exploration experiments are conducted using the aerial platform described in Sect. 5.1 (Fig. 7a) in two outdoor scenarios: (1) Open space (Fig. 7b), and (2) Scattered obstacles (Fig. 7c). Total exploration duration is limited to 90 s to minimize the effects of state estimation drift on the resulting map. During each scenario, the robot explores while confined to a $12\,\text{m} \times 24\,\text{m} \times 2\,\text{m}$ bounding box. The robot starts at the same position in the bounding box for each

Table 3 Performance statistics for the high branching factor, Sim-Time simulation study. Unless otherwise stated, rates refer to average performance over time periods when tracking a given action type. All statistics include any time the robot is tracking a given action, except for the entropy reduction rate in the last row, which is conditional on a significant entropy reduction rate (greater than 600 bits/s). Best (or nearly equivalent) values are bolded

Actions	$\perp$	$\parallel$	Z	Yaw	Stop
Selection frequency	0.343	**0.479**	0.089	0.088	0.001
Total entropy red, norm.	**0.40**	**0.41**	0.06	0.12	0.01
Average speed (m/s)	2.163	**2.778**	0.959	1.114	1.611
Average yaw rate (rad/s)	0.286	0.234	0.170	**0.381**	0.080
Entropy red. rate (bits/s)	**2425**	**2389**	2020	**2414**	1283

trial in both scenarios. The bounds on the speed for perpendicular ($V_{\perp,\max}$) and parallel ($V_{\parallel,\max}$) motions are set at 1.25 m/s and 2.25 m/s, respectively. The explored maps and robot trajectory for two experiments, one from each scenario, are shown in Fig. 7d, e. Speeds achieved by the vehicle during the experiments are shown in Fig. 7f. Figure 8 provides plots of reduction of map entropy as well as summary statistics. Even though the trials were relatively short, the odometry often drifted significantly by the end This drift likely contributed to the robot getting stuck behind an obstacle during the trial **S2**. For this reason, we only use the first for the first 40 s of each trial when computing summary statistics unless otherwise noted.

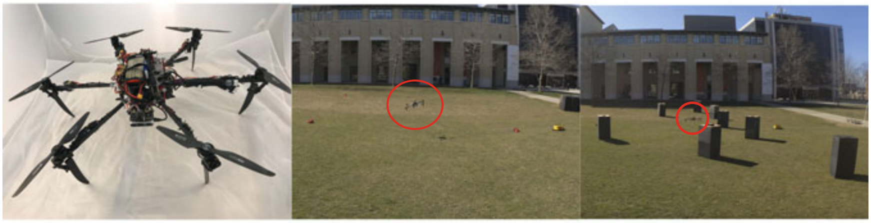
As shown in Fig. 7f, the odometry system reports that the robot reaches and slightly exceeds the maximum desired reference speed[1] in each trial, primarily while executing motions parallel to the frontier. Figure 7d shows a particularly notable example of this behavior where the robot executed an outward spiraling motion soon after the start of the trial.

6 Conclusion and Future Work

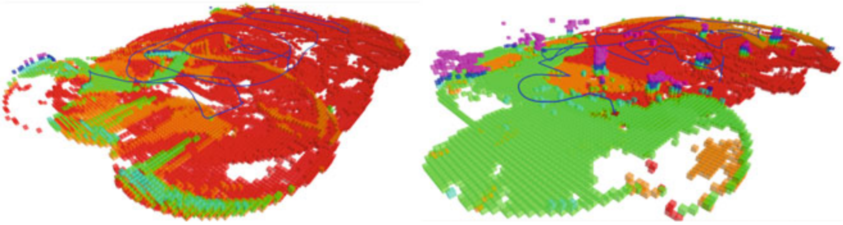
We have investigated how the dynamics of aerial platforms and the geometry of common sensors impact selection of control actions in robotic exploration. We have obtained bounds on the velocity for different kinds of motions and applied this analysis to the design of a motion primitive library and information-based planner suitable for rapid exploration with aerial robots. We have demonstrated this approach

---

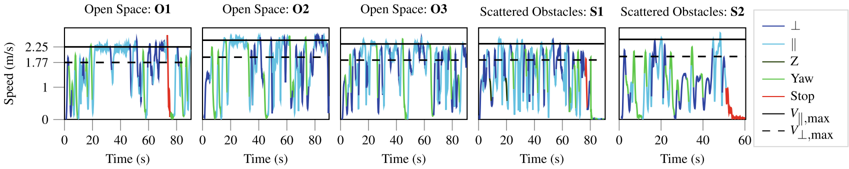
[1] The robot may exceed reference speeds due to an error in tracking the position reference because of environmental disturbances and inaccuracies in the system model.



(a) Multirotor with depth sensor **(b)** Open space scenario **(c)** Obstacles scenario

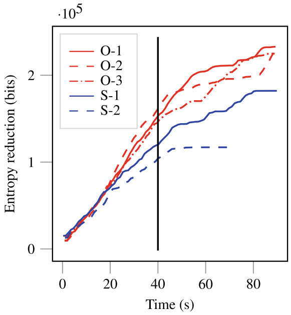


(d) Open space scenario exploration after 90 s **(e)** Obstacles scenario exploration after 90 s



(f) Speeds attained during outdoor exploration experiments

Fig. 7 **a** Multirotor used for hardware experiments in two real-world scenarios: **b** open space and **c** space with scattered obstacles. **d**, **e**, and **f** show the explored maps (color gradient based on Z height), overall paths, and speeds attained by the robot after 90 s of 3D exploration using the proposed exploration approach (Video: https://youtu.be/YXt4yiTpOAc)



Trials:	**O1**	**O2**	**O3**	**S1**	**S2**
Entropy red. at 40s (bits$\times 10^5$)	1.53	1.61	1.46	1.2	1.04
Entropy red. final (bits$\times 10^5$)	2.32	2.27	2.25	1.82	1.16
Average speed (m/s)	1.50	1.56	1.56	1.42	0.98
Average yaw rate (rad/s)	0.39	0.39	0.33	0.33	0.31
Max. speed (m/s)	2.39	2.38	2.38	2.40	2.29
Max. yaw rate (rad/s)	0.66	0.59	0.55	0.60	0.63

Fig. 8 Entropy reduction for hardware trials and summary statistics. Except for the final entropy reduction, all statistics are computed over the first 40 s of each trial (shown by the black bar in the entropy reduction plot)

both in a simulated exploration of a large warehouse environment and in outdoor experiments with only onboard computation. This system produces interesting and intuitive motions in practice such as outward spirals for rapid coverage of open space. Further, the experimental results demonstrate speeds exceeding 2.25 m/s for both open and cluttered environments, which matches and slightly exceeds prior state-of-the-art results [6].

The analysis illuminates competing directions for improvements to speed and entropy reduction performance. Decreasing planning time and latency, such as by improved efficiency or reactive planning [6] or simply increasing sensor range, can improve speeds moving perpendicular to frontiers which may be especially important in highly cluttered environments where motion parallel to frontiers is not viable. At the same time, the ability to safely and rapidly navigate known environments is also tightly coupled to exploration performance both for motions parallel to frontiers and when traversing known space to new unexplored regions. Thus, improvements to state estimation, mapping, and planning under uncertainty are also critical to increasing speed and entropy reduction rates in exploration.

Acknowledgements The authors would like to thank Xuning Yang for help in the discussion and implementation of forward arc motion primitives used in this work.

References

1. Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R.: Receding horizon path planning for 3D exploration and surface inspection. Auton. Robots **42**(2), 291–306 (2018)
2. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–43 (2012)
3. Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., Kumar, V.: Information-theoretic planning with trajectory optimization for dense 3D mapping. In: Proceedings of Robotics: Science and Systems, Rome, Italy (2015)
4. Charrow, B., Liu, S., Kumar, V., Michael, N.: Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In: Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA (2015)
5. Chaslot, G.: Monte-Carlo tree search. Ph.D. thesis, Universiteit Maastricht (2010)
6. Cieslewski, T., Kaufmann, E., Scaramuzza, D.: Rapid exploration with multi-rotors: a frontier selection method for high speed flight. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada (2017)
7. Corah, M., Michael, N.: Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice. Auton. Robots **43**(2), 485–501 (2019)
8. Corah, M., O'Meadhra, C., Goel, K., Michael, N.: Communication-efficient planning and mapping for multi-robot exploration in large environments. IEEE Robot. Autom. Lett. **4**(2), 1715–1721 (2019)
9. Janson, L., Hu, T., Pavone, M.: Safe motion planning in unknown environments: optimality benchmarks and tractable policies. In: Proceedings of Robotics: Science and Systems, Pittsburgh, PA (2018)
10. Kulich, M., Faigl, J., Přeučil, L.: On distance utility in the exploration task. In: Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China (2011)

11. Lauri, M., Ritala, R.: Planning for robotic exploration based on forward simulation. Robot. Auton. Syst. **83**, 15–31 (2016)
12. Liu, S., Watterson, M., Tang, S., Kumar, V.: High speed navigation for quadrotors with limited onboard sensing. In: Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden (2016)
13. Mahony, R., Kumar, V., Corke, P.: Multirotor aerial vehicles: modeling, estimation, and control of quadrotor. IEEE Robot. Autom. Mag. **19**(3), 20–32 (2012)
14. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China (2011)
15. Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The grasp multiple micro-uav testbed. IEEE Robot. Autom. Mag. **17**(3), 56–65 (2010)
16. Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., Tadokoro, S.: Collaborative mapping of an earthquake-damaged building via ground and aerial robots. J. Field Robot. **29**(5), 832–841 (2012)
17. Qin, T., Li, P., Shen, S.: VINS-mono: a robust and versatile monocular visual-inertial state estimator. IEEE Trans. Robot. **34**(4), 1004–1020 (2018). https://doi.org/10.1109/TRO.2018.2853729
18. Spitzer, A., Yang, X., Yao, J., Dhawale, A., Goel, K., Dabhi, M., Collins, M., Boirum, C., Michael, N.: Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor. In: Proceedings of the International Symposium on Experimental Robotics. Springer, Buenos Aires, Argentina (2018) (to be published)
19. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using Rao-Blackwellized particle filters. In: Procedings of Robotics: Science and Systems (2005)
20. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA (1997)
21. Yang, X., Agrawal, A., Sreenath, K., Michael, N.: Online adaptive teleoperation via motion primitives for mobile robots. Auton. Robots (2018)

Ray Tracing and Use of Shadows as Features for Determining Location in Lunar Polar Terrain

Eugene Fang and William “Red” Whittaker

1 Introduction

Solar-powered rovers that explore the poles of the Moon for water ice and other volatiles will rely on sun-synchronous routes to extend a mission from weeks to months [15]. These spatio-temporal routes that chase sunlight and avoid shadows are highly brittle and require accurate localization to ensure rovers are in the right place at the right time. Compared to at equatorial regions, terrain relative navigation is challenged at the lunar poles. Long sweeping shadows preclude views of unlit regions and change the appearance at any location over time as the Sun clocks the diurnal cycle. This causes large errors in current methods for terrain registration, as true rover positions may be interpreted as false negatives due to changing views and appearances.

This research develops techniques to improve the accuracy of terrain relative localization at the lunar poles by considering both near-field and far-field geometry, accounting for sensor and terrain uncertainty, and exploiting shadows as semantic features for registration in an otherwise colorless and homogeneous environment. The true geometry and appearance of terrain influences both the observations from rover stereo imagery as well as orbital 3D models and imagery. Maximizing the likelihood of a rover observation given an expectation based on orbital data yields an estimate of position.

This paper presents a ray tracing method for registering rover stereo imagery to digital elevation models that considers near-field and far-field geometry and takes into account sensor and terrain uncertainty. This method is then augmented to incorporate shadows as features for localization. Section 2 discusses related work in geometric and appearance-based terrain registration. Section 3 presents a ray tracing approach

---

E. Fang (✉) · W. “Red” Whittaker
Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
e-mail: fangeugene@cmu.edu

W. “Red” Whittaker
e-mail: red@cmu.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_22

for geometric terrain registration. An augmentation of the method to exploit shadows as features is described in Sect. 4. Experiments and results are presented in Sect. 5. Section 6 discusses conclusions and directions for future research.

2 Related Work

Common approaches for registering rover imagery to DEMs utilize skyline matching [4, 17]. These methods extract the skyline—the curve where the horizon meets the sky—from a rover image (usually a panorama). Next, the skyline is compared to simulated skylines rendered at different positions within a digital elevation model. The similarity between the detected and simulated skyline provides a likelihood of the rover being at each position.

Accuracy of skyline-based approaches is limited by the effects of parallax. When skyline features are distant—typical for planetary exploration rovers that operate on benign terrain far away from hills and other significantly 3-dimensional terrain features—positions hundreds of meters apart may have very similar skylines. Furthermore, these approaches are especially susceptible to the presence of shadows on the Moon. The black sky due to the lack of atmosphere can be difficult to differentiate from dark shadows (Fig. 1), resulting in an incorrectly detected skyline. Current skyline matching methods do not account for this, and localization accuracy greatly degrades as a result.

Other approaches for terrain registration work by aligning local 3D models generated by onboard sensors to 3D orbital maps. Common variations of this method typically generate local rover point clouds from stereo imagery or LIDAR, downsample them to match the resolution of orbital DEMs, and then find their best alignment to

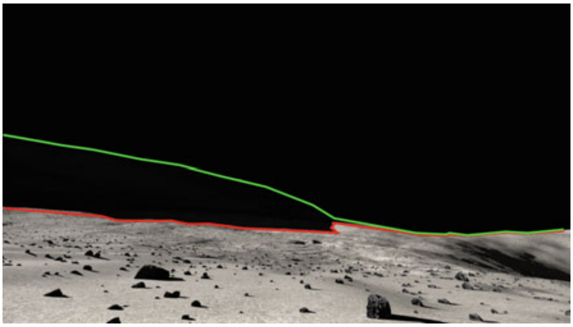


Fig. 1 A rover-perspective render near Shackleton crater on the Moon illustrates how skyline matching approaches for terrain registration are easily confounded by shadows on the Moon. The dark sky is difficult to differentiate from skyline features in shadow, resulting in a detected skyline (red) drastically different from the true skyline (green)

orbital DEMs using variations of the Iterative Closest Point (ICP) algorithm [12, 20] or rock feature extraction [5]. A limitation of stereo methods, however, is the direct relationship between distance and uncertainty in distance. As a result, these methods discard portions of point clouds outside a given radius around the rover where the uncertainty in depth due to disparity error exceeds the resolution of the DEM [13]. However, discarding far-field information reduces the capability of distinguishing the difference between locations that look locally similar and also reduces the size of the basin of attraction for localization.

Methods that use LIDAR or other time-of-flight sensors do not suffer as much from the same issue of uncertainty increasing with distance, so far-field points can also be used. One such method, Multi-frame Odometry-compensated Global Alignment, extracts features from local and global 3D models such as topographic peaks [2] for registration. However, it relies on a high-power surveying-grade LIDAR for centimeter-accurate range measurements up to 1.5 km, which is impractical for non-terrestrial applications where power and mass budgets are much stricter.

Automated registration of photos to prior orbital imagery has been deeply explored for terrestrial aerial vehicles [14, 19] and precise Mars and lunar landings [10, 11]. Many of these methods are all not well suited for surface rovers, where photos taken on the surface differ vastly in scale and have different spatial frequency content from those taken from orbit. Approaches that directly match vehicle images to orbital images require images of the same scale. Those that estimate position using SIFT features fail due to the different level of detail and spatial frequency content.

Some methods attempt to bridge the difference in scales by using a multi-resolution approach [7], but rely on a series of images taken over the length of a traverse in order to accurately localize. These suffer from issues when the terrain local to the rover is barren. Other methods overcome the disparity in scale by only correlating salient features in images such rocks and craters [3, 9], but require that a sufficient number of features that are present and large enough to be visible in orbital imagery.

Even in optimal conditions, all aforementioned approaches suffer from dynamic illumination and shadows which change and obscure features. Synthetic images for registration based on orbital data and simulated lighting could be generated, but this idea has not been explored and the necessary accuracy and resolution of these synthetic images is unknown. Semantic features for localization offers benefits over low-level image features, which suffer from ambiguity and dependence on viewpoint [16]. However, these high-level features are only useful for terrain relative localization if they can be reliably observed or predicted using orbital data.

This research is distinct from prior methods in two important ways. First, it incorporates both near-field and far-field geometry using a ray tracing approach for registering rover-centric stereo imagery to orbital terrain geometry that considers sensor noise and terrain uncertainty. Second, it exploits shadows as semantic features for localization, which handles differences in scale, frequency, and illumination. The developed method enables accurate map-based positioning at the lunar poles where current algorithms fall short.

3 Ray Traced Stereo Scan Matching to Geometric Maps

This section presents a novel method for registering rover stereo images to 3D orbital maps that addresses many of the shortcomings of existing geometric terrain registration methods. Rays that correspond to pixels in stereo images are traced through a 3D map at a candidate pose to evaluate likelihood of observation based on occupancy. The sum of log likelihoods is then maximized through optimization to find the most likely pose within the map given the observations. The method accounts for anisotropic stereo noise to incorporate both near-field and far-field data, considers uncertainty in elevation maps, and explicitly models terrain free-space and occlusion.

3.1 Stereo Correspondence and Ray Construction

The method begins with a pair of undistorted and rectified stereo images captured by a rover. A block matching stereo correspondence algorithm [6] is run to obtain a disparity image (Fig. 2).

The vectors $r_i = (x_i, y_i, z_i)$ that start at the camera origin and pass through each pixel (u_i, v_i) of the disparity image are computed using the camera principal point (c_x, c_y) and focal length f (Fig. 3).

$$x_i = (u_i - c_x)/f$$

$$y_i = (v_i - c_y)/f$$

$$z_i = 1$$

These vectors, or *rays*, form the basis for individual observations that will be evaluated for the likelihood of occurrence.

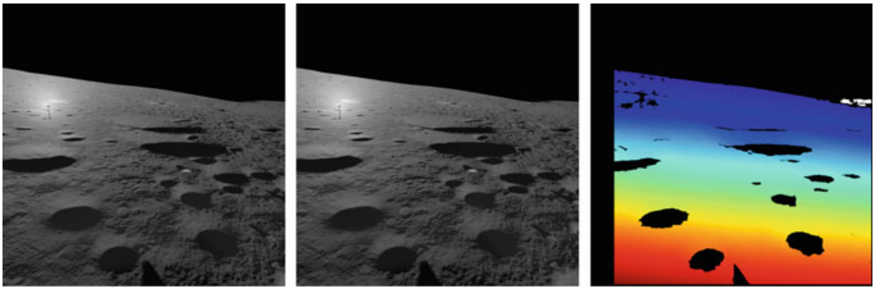

(a) Left stereo image (b) Right stereo image (c) disparity image

Fig. 2 A disparity image **c** constructed from a left **a** and right **b** stereo image pair. Red indicates large disparity, blue indicates small disparity, white indicates zero disparity, and black indicates no stereo match

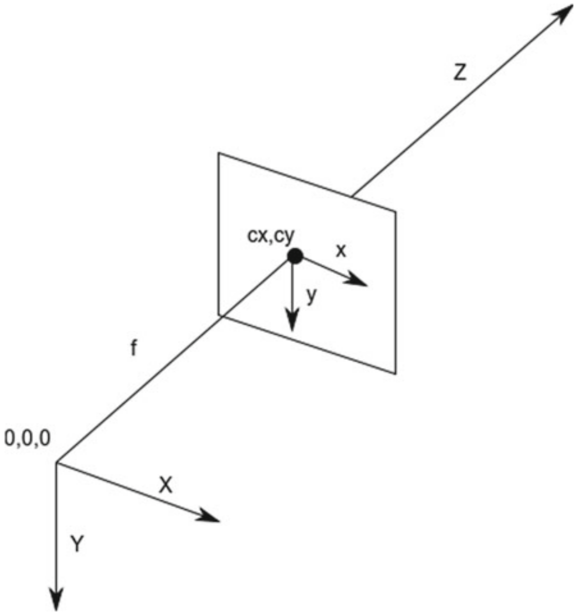


Fig. 3 Camera coordinate system

3.2 Camera Sensor Model Based on Disparity

Each traced ray from the disparity image (Fig. 2c) comes from a pixel with either (a) non-zero disparity, (b) zero disparity, or (c) no stereo match due to poor features or lack of features. To calculate depth and uncertainty for each type of observation, they must be handled separately.

3.2.1 Non-zero Disparity

Pixels with non-zero disparity typically make up the majority of a disparity image. For standard rectified stereo geometry, the inverse relationship between depth Z and disparity d is given by

$$Z = \frac{fB}{d}$$

where f is the focal length and B is the stereo baseline [18]. Due to this non-linear inverse relationship, Gaussian error in pixel-space disparity results in non-Gaussian, asymmetric error in depth (Fig. 4). A plot of the likelihood of occupancy at a depth for different observed stereo disparities is shown in Fig. 5. Values of $f = 50px$, $B = 0.5\,\mathrm{m}$, and $\sigma_{disparity} = 0.1px$ are chosen to exaggerate the effect of stereo disparity for illustrative purposes. Large disparities correspond to smaller depths with lower uncertainty. Small disparities correspond to larger depths with higher uncertainty.

Fig. 4 Gaussian error in disparity d results in non-Gaussian, asymmetric error in depth Z. *Image credit* [8]

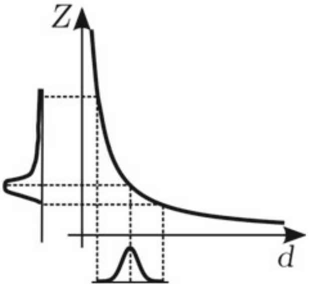
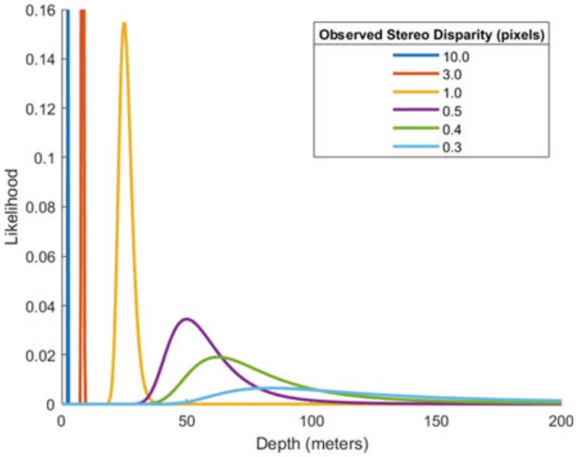


Fig. 5 As the observed stereo disparity decreases, the likely depth and uncertainty in depth both increase



3.2.2 Zero Disparity

For many registration algorithms, such as those that rely on point clouds, pixels with zero stereo disparity cannot be used because they provide no information about depth. However, with ray tracing, they provide a lower bound for which the likelihood of occupancy at a depth—though possibly infinitesimally small—is non-zero. To overcome the issue of infinitesimally small likelihoods rounding to zero given the finite precision of computing, rays are only traced out to a finite range. Figure 6 shows an example of the likelihood of depth given an observation with zero stereo disparity up to a range of 2500 m.

3.2.3 No Stereo Match

In cases where there is no stereo match due the lack of features, registration relies on semantic scene classification. This idea is explored in Sect. 4.

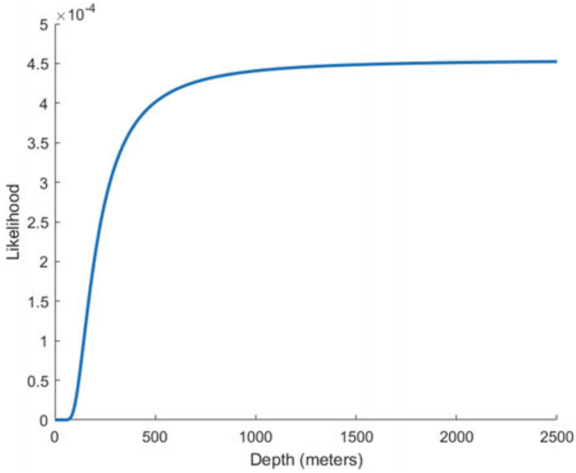
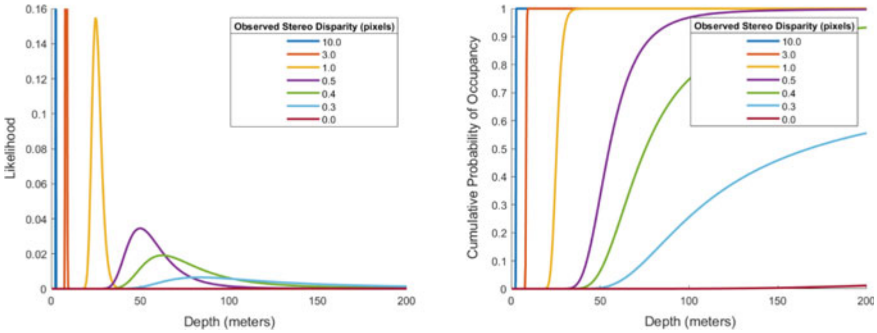


Fig. 6 The likely depth for an observation with zero stereo disparity. Range is limited to 2500 m



(a) Model of occupancy likelihood as a function of depth for several observed stereo disparities

(b) Model of cumulative likelihood of occupancy for several observed stere disparities

Fig. 7 The probability that an observed stereo disparity is occupied before a given depth is equal to the cumulative likelihood of occupancy

3.3 Observed Terrain Occupancy

In order for the evaluation of observation likelihood that accounts for occlusion, the raw likelihood of occupancy as a function of depth is not used. Instead, the cumulative likelihood of occupancy is computed (Fig. 7). This gives the probability that a observed ray with a particular stereo disparity is occupied before a given depth along that ray, which enables occlusion-aware observation likelihood maximization.

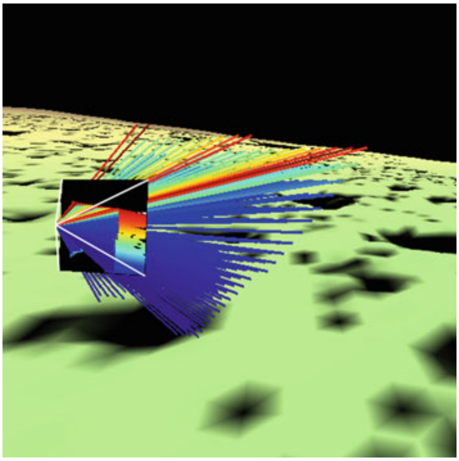


Fig. 8 Given a candidate pose in an elevation map, rays are traced within the elevation map to find the expected cumulative likelihood of occupancy along each ray

3.4 Expected Terrain Occupancy

To evaluate the likelihood of a ray observation, the observed and expected cumulative occupancy likelihood along the ray are compared. The expectation is computed by tracing the ray in an elevation map at a candidate pose (Fig. 8). Due to the nature of ray tracing, computing the expected occupancy can incorporate uncertainty in the elevation map. Marching along a ray, the probability of occupancy is given by

$$P(o_r|e_r) = cdf(o_r|\mu(e_r), \sigma(e_r))$$

where o_r is the elevation at range r, e_r is the cell in the elevation map that contains o_r, μ and σ are the mean and variance of the elevation measurement, and cdf is the cumulative probability distribution function.

3.5 Observation Likelihood Maximization

$P(o_i|e_i)$ is the likelihood of obtaining a single ray observation o_i given an expected observation e_i for ray i, and is calculated by computing the inverse absolute area between the observation and expectation curves (Fig. 9).

Summing the log likelihood across all rays yields the overall likelihood.

$$\mathscr{L} = \sum_i \log(P(o_i|e_i))$$

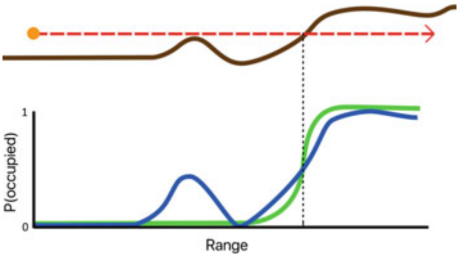


Fig. 9 A ray (dotted red) traced from a candidate pose (orange) intersects the terrain (brown). The observed probability of occupancy along the ray is shown in green, and the expected probability of occupancy along the ray is shown in blue (assuming some terrain uncertainty). The likelihood of the observation given the expectation is scored as the inverse absolute area between the two curves

Nelder-Mead optimization is used to maximize the likelihood over a candidate pose. If initialized close enough to the true pose, the estimate of the rover's 3D pose converges to the true pose. Experiments in Sect. 5 explores the performance of this optimization.

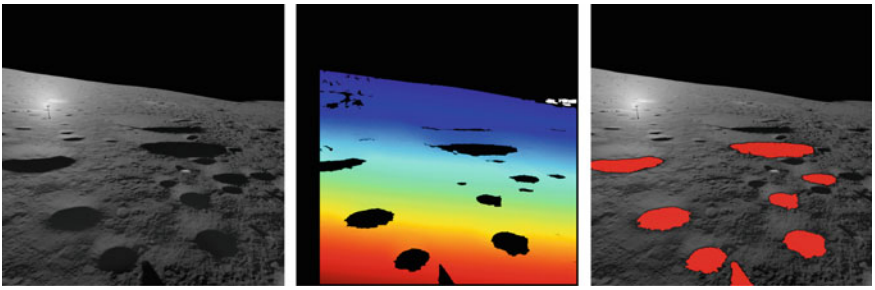
4 Detection and Registration of Shadow Features

Pure geometric methods for terrain registration fail when local topology is not distinct. Appearance-based terrain registration succeeds when geometric methods do not by correlating colors, textures, and other visual features to prior orbital imagery. However, appearance-based methods are severely limited at the lunar poles where there is no distinguishing color, textures depend on illumination, and visual features may be occluded by shadow. Furthermore, correlating photos taken by a surface rover to those taken by an orbiting satellite is generally challenging, since the photos differ vastly in vantage. These limitations are compounded by the constantly varying lighting and shadows.

Instead of being hindered by shadows, this research explores the idea of modeling them over time and exploiting them as features. This section presents a novel use of shadows as features for terrain registration that overcomes the vantage challenge of appearance-based registration by relying on semantic features rather than raw image features, seamlessly incorporates shadow registration into the ray traced stereo scan matching algorithm, and demonstrates its advantages in simulated experiments.

4.1 Shadow Detection

Semantic detection of shadows in an image provides features for terrain registration, and succeeds even in image areas with no features for stereo correspondence (Fig. 10).



(a) Left stereo image

(b) Disparity image. Black ateas have no stereo correspondence.

(c) Detected shadow features (red)

Fig. 10 Regions in stereo imagery (**a**) with no features for stereo correspondence (**b**) can still provide semantic features for registration (**c**)

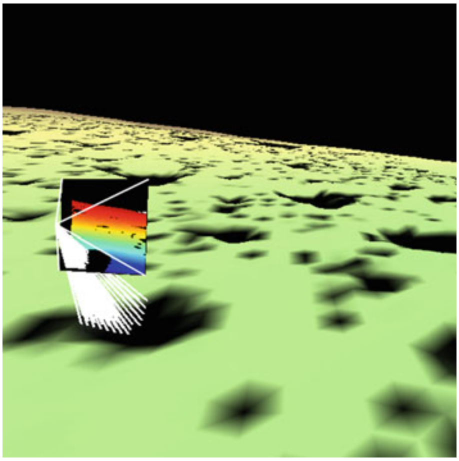


Fig. 11 Given a candidate pose in an elevation map, rays corresponding to semantically detected shadows (white) are traced within the map to find the likelihood of the ray intersecting an area of terrain in shadow

The early approach for this research for automated detection of large shadows uses an adaptive threshold on intensity and then filters those by the size of connected components.

4.2 *Shadow Registration Using Ray Tracing*

Since the ray traced stereo scan matching algorithm described in Sect. 3 minimizes the log likelihood of an observation o_i given an expected observation e_i, incorporation of shadow features into the registration process requires only a probability model $P(o_i|e_i)$ for each ray corresponding to a shadow. With the current naive approach for shadow detection, o_i is binary. Thus, $P(o_i|e_i)$ is given by the probability that a shadow ray intersects a given patch of terrain multiplied by the probability of that patch of terrain being in shadow. Figure 11 shows an illustration of rays corresponding to shadows being traced within an elevation map.

5 Experiments and Results

Ray traced scan matching for terrain registration was evaluated against other methods in a photo-realistic lunar simulator developed by NASA Ames Research Center and Open Robotics [1] (Fig. 12). The simulator covers an area of 1.2 × 1.2 km and uses a DEM by Nobile Crater near the south pole of the Moon. Fractal expansion was used to upsample the DEM to a resolution of 4 cm/pixel in a small 160 × 160 m region at the center, and gradually decreasing resolution outwards. The rover's stereo cameras have a resolution of 1024 × 1024 pixels, a 70° field of view, and a stereo baseline of 40 cm.

The simulated rover was commanded to take an overlapping panorama of six pairs of stereo images centered 15° below the horizon. Different terrain registration algorithms used these images as input and were randomly initialized at 10 locations up to 10 m away from the true position. After attempting registration from each of these initialization conditions, the residual error with the best convergence score was then recorded. This process was repeated over a 10 m grid within the high-resolution 160 × 160 m portion of the simulated environment, and the results are shown in Figs. 13 and 14. On average, compared to ICP, the developed methodology decreases the average error from 0.5 to 0.15 m and decreases the maximum error from 3.10 to 0.71 m.

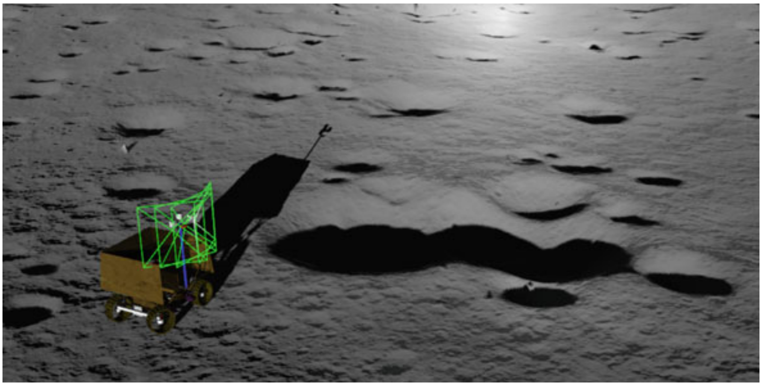


Fig. 12 A screenshot from NASA's planetary rover simulator for lunar missions showing a rover, its cameras' field of views, and the surrounding terrain

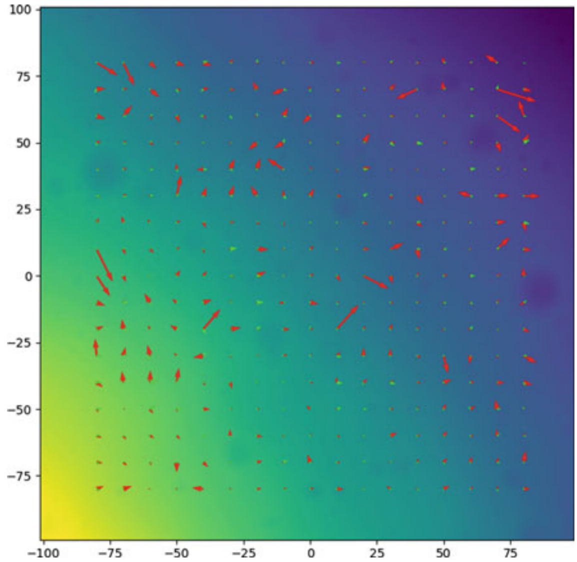


Fig. 13 The residual localization error (scaled 5x for clarity) at 289 locations within the simulated environment is compared. Point-to-plane ICP (red) has significantly larger errors than ray traced scan matching (green)

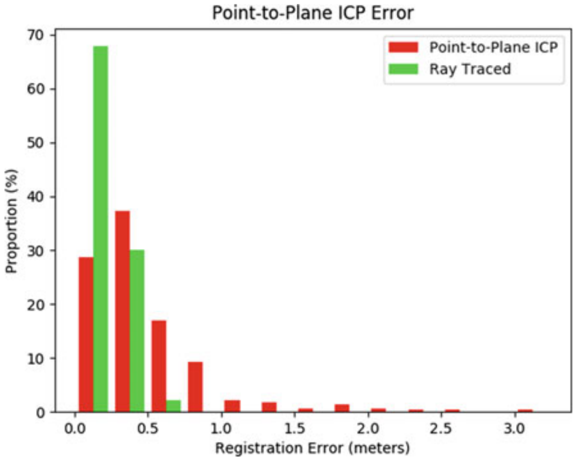


Fig. 14 In this particular dataset, ray traced scan matching (green) decreases both the mean and maximum registration error when compared to point-to-plane ICP (red)

Geometry-only and geometry-and-shadow ray traced stereo scan matching were compared at 289 locations within the same 160 × 160 m region of the lunar simulator. For each method at each location, the likelihood field on a 20 × 20 m grid around the true position in x-y is computed. For reliable optimization, the desired shape of these likelihood fields is a continuous, steep slope toward the true optima. Figure 15 compares the likelihood fields of geometry-only and geometry-and-shadow scan matching over the entire 160 × 160 m region with a fixed color scale. Qualitatively, the addition of registering shadow features seems to decrease positional uncertainty overall. Quantitatively, by computing the volume under the likelihood fields, the addition of shadows decreases the volume by 26% on average, and never increases it.

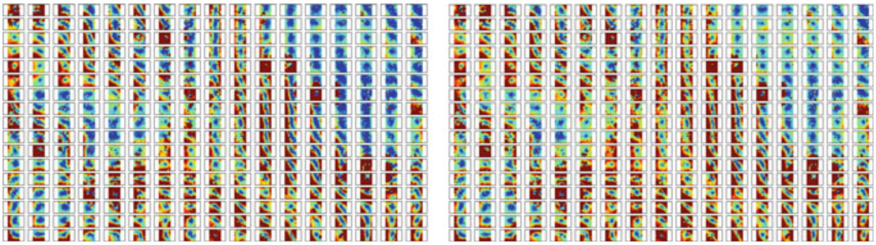


Fig. 15 Geometry-only (left) and geometry-and-shadow (right) likelihood fields around true positions for 289 locations in a 160 × 160 m region of the lunar simulator. Color scales fixed, with blue as high-likelihood and red as low-likelihood. Overall, the addition of shadows decreases positional uncertainty and never increases it

Using a single core on a multi-core Intel i7 laptop processor from 2011, ray tracing registration took on average 2.9x longer than ICP to converge (19.2 vs. 6.6 s) for each initialization in these experiments. Despite the comparatively worse performance, this is still reasonable for lunar polar sun-synchronous routes that can require maximum driving speeds as low as 1 cm per second [15].

6 Conclusion and Future Work

Sun-synchronous routes could be the key to lunar water if robots could reliably and autonomously navigate them. These routes are delicate, and requires certainty of location at given times. This research developed an approach to accurately localize to the terrain at the lunar poles. In contrast to priors, this method utilizes near-field and far-field geometry and effectively utilizes shadows as features for registration, yielding a reduction in maximum registration error by a factor of four compared to ICP in simulated experiments when initialized within 10 m of the true position. Future work will analyze the efficacy of this method with real field data.

Acknowledgements This work was partially supported by the NASA Space Technology Research Fellowship under grant NNX16AM68H.

References

1. Allan, M., Wong, U., Furlong, P.M., Rogg, A., McMichael, S., Welsh, T., Chen, I., Peters, S., Gerkey, B., Quigley, M., et al.: Planetary rover simulation for lunar exploration missions. In: 2019 IEEE Aerospace Conference, pp. 1–19. IEEE (2019)
2. Carle, P.J.F., Furgale, P.T., Barfoot, T.D.: Long-range rover localization by matching lidar scans to orbital elevation maps. J. Field Robot. **27**(3), 344–370 (2010)
3. Cheng, Y., Johnson, A.E., Matthies, L.H., Olson, C.F.: Optical landmark detection for spacecraft navigation (2003)

4. Cozman, F., Krotkov, E., Guestrin, C.: Outdoor visual position estimation for planetary rovers. Auton. Robot. **9**(2), 135–150 (2000)
5. Di, K., Liu, Z., Yue, Z.: Mars rover localization based on feature matching between ground and orbital imagery. Photogramm. Eng. Remote Sens. **77**(8), 781–791 (2011)
6. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. Int. J. Comput. Vision **70**(1), 41–54 (2006)
7. Foil, G., Cunningham, C., Wettergreen, D.S., Whittaker, W.L.: Onboard detection and correction of orbital registration errors using rover imagery (2014)
8. Gutiérrez-Gómez, D., Mayol-Cuevas, W., Guerrero, J.J.: Inverse depth for accurate photometric and geometric error minimisation in rgb-d dense visual odometry. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 83–89. IEEE (2015)
9. Hwangbo, J.W., Di, K., Li, R.: Integration of orbital and ground image networks for the automation of rover localization. In: ASPRS 2009 Annual Conference (2009)
10. Johnson, A.E., Cheng, Y., Montgomery, J.F., Trawny, N., Tweddle, B., Zheng, J.X.: Real-time terrain relative navigation test results from a relevant environment for mars landing. In: AIAA Guidance, Navigation, and Control Conference, p. 0851 (2015)
11. Johnson, A.E., Montgomery, J.F.: Overview of terrain relative navigation approaches for precise lunar landing. In: 2008 IEEE Aerospace Conference, pp. 1–10. IEEE (2008)
12. Nefian, A.V., Bouyssounouse, X., Edwards, L., Kim, T., Hand, E., Rhizor, J., Deans, M., Bebis, G., Fong, T.: Planetary rover localization within orbital maps. In: 2014 IEEE International Conference on Image Processing (ICIP), pp. 1628–1632. IEEE (2014)
13. Nefian, A.V., Edwards, L.J., Lees, D., Keely, L., Parker, T.J., Malin, M.: Automatic rover localization in orbital maps. In: Lunar and Planetary Science Conference, vol. 48 (2017)
14. Oh, J., Toth, C.K., Grejner-Brzezinska, D.A.: Automatic georeferencing of aerial images using stereo high-resolution satellite images. Photogramm. Eng. Remote Sens. **77**(11), 1157–1168 (2011)
15. Otten, N.D.: Planning for Sun-Synchronous Lunar Polar Roving. Ph.D. thesis (2018)
16. Rogers, J.G., Trevor, A.J.B., Nieto-Granda, C., Christensen, H.I.: Simultaneous localization and mapping with learned object recognition and semantic data association. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1264–1270. IEEE (2011)
17. Stein, F., Medioni, G.: Map-based localization using the panoramic horizon. IEEE Trans. Robot. Automat. **11**(6), 892–896 (1995)
18. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer Science & Business Media (2010)
19. Wong, A., Clausi, D.A.: Arrsi: Automatic registration of remote-sensing images. IEEE Trans. Geosci. Remote Sens. **45**(5), 1483–1493 (2007)
20. Woods, M., Shaw, A., Tidey, E., Van Pham, B., Simon, L., Mukherji, R., Maddison, B., Cross, G., Kisdi, A., Tubby, W., et al.: Seeker-autonomous long-range rover navigation for remote exploration. J. Field Robot. **31**(6), 940–968 (2014)

Planning for Robotic Dry Stacking with Irregular Stones

Yifang Liu, Jiwon Choi, and Nils Napp

1 Introduction

We critically depend on modifying our environment, e.g., by constructing shelters, infrastructure for transportation, water, energy, and waste management, as well as structures that regulate the natural environment such as dams, drainage, and protection against avalanches. Labor and materials are the main cost drivers of the construction industry, which also produces approximately 500 million tons of demolition waste, mostly in the form of concrete [1]. Cement production accounts for (≈%5) of global CO_2 emission [2]. Robotic construction with in situ (found) materials simultaneously addresses primary cost drivers of construction, while mitigating its environmental impact. This idea has been explored in specialized situations. Driven by the need for resource conservation in space, NASA has studied in situ material use for extraterrestrial environments. Launching building materials into space is very costly, yet simple structures–such as berms, walls, and shelters–might be readily built from minimally processed but rearranged materials [3]. Such *utility structures*, i.e., structures that have a specific function, but whose exact shape matters less, are also important on Earth. Examples include erosion barriers for changing coastlines, temporary support structures in disaster sites, or containment structures made from contaminated materials from a nuclear or chemical leak. One particularly well-suited

---

Y. Liu (✉) · N. Napp
School of Electrical and Computer Engineering, Cornell University,
Ithaca, NY 14853, USA
e-mail: yl892@cornell.edu

N. Napp
e-mail: nnapp@cornell.edu

J. Choi
Department of Computer Science and Engineering, University at Buffalo,
Buffalo, NY 14260, USA
e-mail: jiwoncho@buffalo.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_23

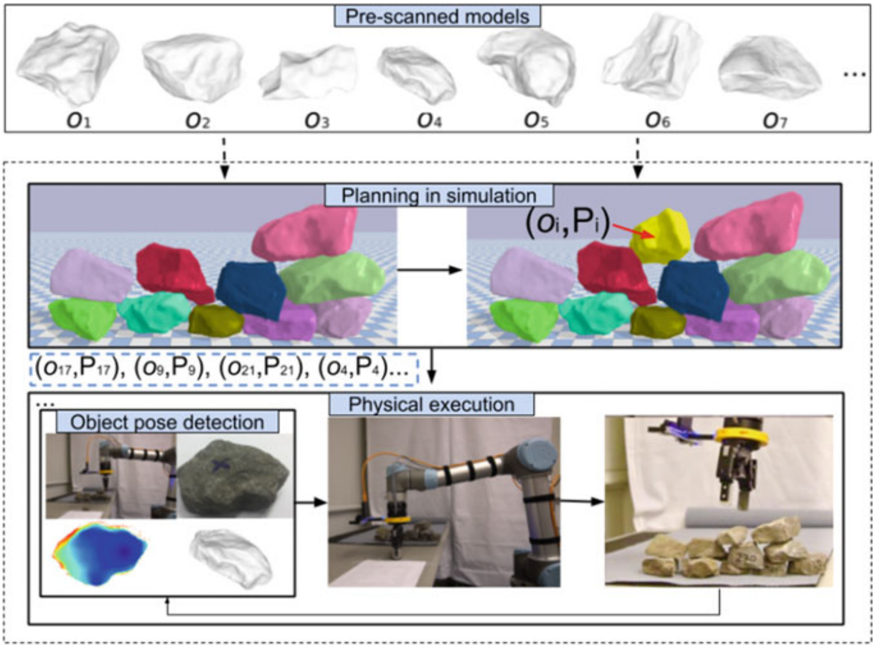


Fig. 1 System overview. All the stone models are pre-scanned. The planning is conducted in a physics simulator, then the assembly sequences are executed open-loop with a robotic arm

construction method for such types of utility structures is dry-stacking stones. This ancient method was practiced by humankind since 3500 B.C. [4], and makes up some of the oldest man-made structures. Theoretically, robots are ideally suited for this work, since robots make work safer and physically less demanding. A lack of understanding of how to pose and solve assembly planning problems of irregular natural material into in situ functional structures, however, is currently hindering robots from performing such useful construction tasks.

We present an algorithmic approach for solving the planning problem of assembling stable structures from a collection of irregularly shaped rigid objects. The application is to enable dry stacking with found, minimally processed rocks. We focus on the problem of high-level placement planning for rocks to build stable structures, and dry-stack structures with tens of rocks, which significantly improves the state of the art. These plans are executed open-loop without additional tactile sensing; however, our results suggest that large-scale dry-stacking robots would benefit from better physical feedback during the construction process. The whole process is shown in Fig. 1. To the best of our knowledge, this is the first system that can automatically dry-stack a wall with 4 courses using natural irregular rocks, which could significantly benefit the large-scale outdoor construction robots.

The rest of the paper is structured as follows: Sect. 2 provides a brief overview of related work; Sect. 3 describes the planning algorithm; results are presented in Sect. 4. Finally, Sect. 5 concludes the paper and discusses future work.

2 Related Work

The fundamental questions in autonomous construction are how to specify, plan, and execute the placement of building elements to achieve a final structure. Approaches [5–8] differ in each of these aspects, and range from determining the assembly order of elements whose position is known in advance [5] to formulating building plans that need to pick the type, shape, and pose of elements to build approximate shapes, or to build structures that fulfill specific functions [9].

There are many examples of multi-robot construction [10]. Prominent examples include 3D construction according to blueprints with climbing [11] and flying [12] robots. These systems rely on uniform, custom-made rigid elements that snap together using magnets, and focus on combinations of pre-processed compilation steps and local runtime control. Another body of work is focused on distributed collective robotic construction of functional structures composed of deformable and amorphous materials [9, 13–16]. In these demonstrations, the deformable nature of the building material compensates for placement inaccuracies and environmental irregularity which simplifies planning.

When building with rigid irregular objects, small surface features substantially affect the friction and stability. Compounding this difficulty, microfracture formation during execution may deform the structure surface. This makes planning and stable placement of irregular objects fundamentally different from building with regular objects that have predictable contact geometry. In our previous work [8, 17, 18], we proposed an architecture for solving the dry-stacking problem, based on heuristics and deep Q-learning to build stable, large-scale structures using physics simulation in 2D.

We leverage guidelines for building a stable structure from instructional books for dry-stacked masonry [19–21]. For example, it is good to place large stones with inward-sloping top surfaces on the corners, and smaller stones in the middle. Such heuristics can provide a structured approach in making assembly decisions, but in their description, much is left to experience and human judgment.

The most closely related work is Furrer et al. [6], who propose a pose search algorithm that considers structural stability using a physics simulator. In addition, they present an autonomous system, using a robot manipulator, for stacking stable vertical towers with irregular stones. The pose search cost function considers support polygon area, kinetic energy, the deviation between thrust line direction and the normal of the support polygon surface, and the length between the new object and the center of mass (CoM) of the previously stacked object. While this paper also uses a physics simulator, we use it primarily to find a finite set of feasible poses, and then apply a layered filtering approach, which we found results in both better and more robust performance.

3 Methods

In this section, we first describe the notation used in this paper; then we elaborate on the planning algorithm for stacking irregular stones. Finally, we provide the object pose detection pipeline used in physical execution.

3.1 Notation

The *world frame* is a 3D coordinate system where the gravity is in the negative z-direction, and the goal is to construct a target structure $T \subset \mathbb{R}^3$, i.e., a subset of the world space that should be filled by selecting and placing elements from a set of objects O. Each object is a connected subset of $\mathbb{R}^3$ with its origin at the CoM.

An assembly $A = (a_1, a_2, ..., a_I)$ is a set of I assembled objects, where each element $a_i = (o_i, P_i)$ is a pair containing an object $o_i \in O$ and its pose $P_i = (p_i, R_i) \in SE(3)$. The position $p_i \in \mathbb{R}^3$ denotes the CoM position of object o_i in the world frame, and $R_i \in SO(3)$ is its orientation. Empty space set is a set $E \subset T$ s.t. every point $e \in E$ can be connected by a straight line from ∂T to ∂A without passing through any other a_i and ∂T, and ∂A denote the boundaries of T and the assembly, respectively. This definition excludes the complicated internal voids created by stacking irregular objects from counting as empty space. The top surface is given by $S = \partial E \cap \partial A$, i.e., the overlapping area of the empty set E and the assembly A, where ∂E denotes the boundaries of empty space E. We define the action space $X(o_i)$ of object o_i to be restricted to have its CoM in E:

$$X(o_i) = \{(p_i, R_i) | p \in E\}. \tag{1}$$

The world is initially assumed to be empty of objects, aside from a support surface at the bottom of T. We want to find an assembly strategy for autonomous agents, i.e., picking a sequence of elements o_i and actions from $X(o_i)$ to build an assembly that occupies the target structure T subject to physical contact, friction, and gravity constraints.

3.2 Structure Planning

This section presents an assembly planning algorithm for irregular objects. Similar to [8], we design a greedy heuristic approach to find the next best pose from a set of feasible poses for a given object. For each object, we use a physics simulator to generate a finite set of feasible *stable* poses, strategically reduce this set, and choose the best available pose. By repeating this sequence, we incrementally build the structure.

Algorithm 1: Feasible Pose Generation

```
Data: $o_i$: object
Result: Feasible Poses Set
$\{(x_j, y_j, z_j)\} \leftarrow$ discretize position;
for each (x,y,z) in $\{(x_j, y_j, z_j)\}$ do
    for $N_{ori} \leftarrow 0$ to $N$ do
        $R \leftarrow$ random generate an orientation;
        reset $o_i$ pose to $(x, y, z, R)$;
        while $N_{contact}(o_i) < 3$ do
            step physics simulation once;
        end
        pause physics simulation;
        reset $o_i$ linear and angular velocity to 0;
        step physics simulation once;
        while $o_i$ is not stable do
            step physics simulation once;
        end
        if distance between current pose $P_i$ and $(x, y, z, R) <$ Threshold then
            add current pose $P_i$ to Feasible Pose Set $X_F$;
        else
            continue;
        end
    end
end
```

3.2.1 Feasible Pose Generation

We use a physics simulator to find *physically stable* configurations. We approximate the real-world state with simulation and provide a practical and efficient stability estimate of the system without actually having to physically interact with the external world. This helps us to acquire a good prior estimate for the system.

Since the action space (p and R) is continuous (Eq. 1), we first sample the action space in such a way that the position p is discretized, and each position p corresponds to a set of randomly sampled orientations R. We then make use of a rigid body simulator to find physically stable configurations. The simulator proceeds by first selecting an initial placement (position and orientation) for a given object on the surface of the built structure, and then simulating the forces acting on the object until it settles into a stable pose; see Algorithm 1. Although the number of possible initial placements is large, a substantial amount of them settles down into a small subset of feasible poses. This set of feasible poses for an object o_i denotes $X_F(o_i)$. Even though all the poses in the feasible pose set are stable, many of them are poor choices and result in low overall stability. In the next section, we will discuss how to refine this set by using heuristics gathered from instructional literature for masonry books [19–21].

3.2.2 Action Space Reduction

The refinement of action space is a hierarchical filtering approach, where each filter removes poses that do not meet the minimum requirement for a satisfactory placement according to a specific heuristic. The set of filters used in this work is presented as follows:

- Support polygon area: the area of an object's support polygon. A higher value of support polygons correlates to a stable footing for the object. Similar to the method presented in [6], we robustly find the support polygon from the sparse contacts by updating the simulator 10 steps and collecting all the contacts. Then, Principal Component Analysis (PCA) is used to reduce the 3D contacts to 2D points. Finally, the convex hull of these 2D points is calculated as a support polygon area.
- Normal of support polygon: the normal direction of the support polygon. It measures how much the normal direction deviates from the thrust line direction vector [6].
- Neighbor height: the difference in heights of the object, after its placement, with its left and right neighboring objects. The height of an object is represented by CoM height. This feature helps maintain leveled surfaces in the structure.
- Stone top surface slope: the top surface angle of the object at a given pose. In building a wall, we prefer inward sloping angles to prevent stones from the top layers to fall down from the structure [21, Pg. 49].
- Interlocking: the number of objects in the structure that are in contact with the current object at a given pose. The use of this feature allows for staggered layering and thus helps to prevent vertical stacking in the structure [20, Pg. 19].

Each filter evaluates one of these features. The reductions are applied hierarchically for each object as follows:

- The original feasible pose set denotes X_F.
- At *Filter 1*, only select poses that have an inward sloping top surface angle. The remaining poses after applying this filter denote X_{F1}.
- At *Filter 2*, discard poses with dot product $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$ less than the mean of all poses from X_{F1}, where $\mathrm{n_i}$ represents the normal direction of the contact polygon, and $\mathrm{v_i}$ is the thrust line direction vector. The set of poses after applying this filter denotes X_{F2}.
- At *Filter 3*, remove poses with support polygon area less than the mean of the support polygon area value of all stable poses. The remaining poses after applying this filter denote X_{F3}.

- At *Filter 4*, this is based on the current state of the structure. If it is not a corner placement, we only choose poses whose centroid heights are lower than the average centroid heights of corner stones at the current course. The set of poses after applying this filter denotes X_{F4}.
- At *Filter 5*, remove the poses whose number of interlocking objects are smaller than the mean number of interlocking. The set of poses after applying this filter denotes X_{F5}.

This hierarchical reduction model is carefully designed such that a random pose at each level is more desirable than a random pose drawn at the earlier filtered levels. It is also designed such that no good possible stable poses are removed earlier before reaching the final selection filter. The relation between the various sets of poses is shown in Eq. 2.

$$X_F \supset X_{F1} \supset X_{F2} \supset X_{F3} \supseteq X_{F4} \supset X_{F5}, \tag{2}$$

where X_F is defined in Sect. 3.2.1.

Unlike the pose searching algorithm used in [6], which combines terms similar to Filters 2 and 3, as well as other heuristics into a single scalar cost function and finds poses by gradient descent, the planning algorithm proposed in this paper first considers geometric and physical constraints using a simulator to find a discrete set of feasible actions and further refines this set by using a hierarchical filter based on heuristics gathered from the instructional materials. This approach eliminates the need for tuning the relative weights in a scalar cost function. Without the need for cost-tuning, the algorithm is more adaptable to different stones with various physical properties, such as size, density, and friction. The reason is that with a single scalar function the weights are coupled and the relative importance depends on the set of objects's physical properties. However, in the hierarchical filter, each term is assessed in isolation and thus the method is less sensitive to the change of physical properties in the set of objects.

3.2.3 Proposed Algorithm

Algorithm 2 describes how a structure is constructed. The inputs are the set of available objects along with the target structure to be built. During construction, it builds the structure course by course, and within courses, it first places the corner stones with the inward slope in the two course extrema, as shown in Algorithm 2 Line 2; then it builds the middle area within the course (Lines 3–5). The output is the set of assembly steps.

Algorithm 3 describes the steps to select an object and its pose for the placement. The inputs are the set of remaining objects and their type (corner stone or random stone), since different object types may require different hierarchical filters. The first step is to choose a random object (Line 3) and collect the feasible poses (Sect. 3.2.1) of this object (Line 4). Then, it applies the Hierarchical Filter (Sect. 3.2.2) to reduce the action space at Line 5. If the reduced action space is not empty, we select one pose from it; otherwise, we try this procedure again for a different object until it reaches the maximum number of trials (Lines 6–11).

Algorithm 2: Proposed Assembly Approach

```
   Data: $O$: object dataset, $T$: target structure
   Result: Assembly steps
 1 while target area $T$ still has room left to build do
 2     place corner stones with inward slope in the two course extrema;
 3     while current course still has room left to build do
 4         place stone in the current course ;
 5     end
 6 end
```

Algorithm 3: Place Stone

```
   Data: $B$: set of available objects ($B \subseteq O$), object type
   Result: Placed object pose
 1 $n \leftarrow 0$;
 2 while $n \leq$ Maximum Number of Trials do
 3     $b \leftarrow$ randomly choose one object from $B$;
 4     $X_F \leftarrow$ feasible poses set;
 5     $X_{final} \leftarrow$ apply Hierarchical Filter to $X_F$ ;
 6     if $X_{final} \neq \emptyset$ then
 7         place one of the $X_{final}$ poses;
 8         return;
 9     else
10         $n \leftarrow n + 1$;
11     end
12 end
```

3.3 Object Pose Detection

During physical execution, we first detect the object pose in the scene. We start by capturing a set of point cloud data of an object from different views via an RGB-D camera; we then filter out the points that do not belong to the current object by removing the plane points from point cloud data using Point Cloud Library (PCL) [22]; next, we merge the remaining point cloud data. We apply global registration to provide an initial transformation and Iterative Closest Point (ICP) algorithm to further refine the transformation using the Open3D library [23]; finally, we run registration on merged point cloud data and pre-scanned 3D mesh of the object to get the relative pose between them. Similar to the third step above, the registration also contains global registration and ICP. The whole pose detection pipeline is shown in Fig. 2.

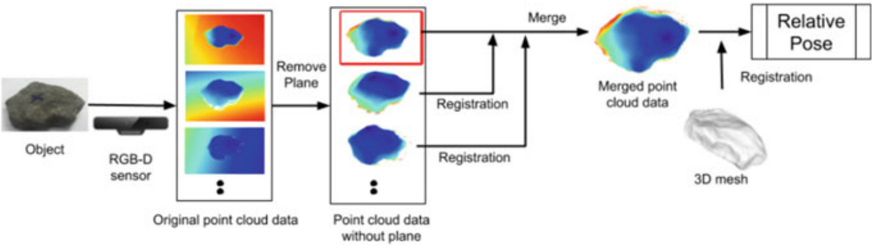


Fig. 2 Object pose detection pipeline showing the sequence of detection, registration, and merging. See Sect. 3.3 for more details

Once the relative pose between the current object and the 3D mesh is detected, we use the manipulator to pick up the stone and apply the same transformation to the end-effector of the manipulator to place the stone as the planned pose.

4 Experiments

In this section, we first describe the experimental setup, then we show the stone towers and stone walls using the proposed planning, as well as the comparison between the pose searching algorithm proposed in [6] and the proposed method.

4.1 Experimental Setup

As shown in Fig. 4, a UR5 manipulator equipped with a ROBOTIQ 2-Finger gripper is used in the manipulation task. An Intel® RealSense™ SR300 RGB-D camera is attached to the UR5 arm for point cloud data acquisition. The MoveIt! [24] package is employed for motion planning. We collected 23 shale stones as irregular objects, which are specifically selected such that they fit the size of the gripper. The average weight of the selected stones is 193 g with a standard deviation of 90 g. The outer bounding box size of the stones are 0.0791 ± 0.0144, 0.0585 ± 0.0086, and 0.04 ± 0.0088 m. During physical execution, objects are manually fed into the pickup area. The gripper grasping position varies depending on the detected object position, but the gripper orientation remains the same. The stone 3D model is acquired with a Matter and Form 3D Desktop Scanner. Figure 3 shows some samples of the stones and their corresponding 3D mesh models. The object pose detection and manipulation parts are implemented using the Robot Operating System.

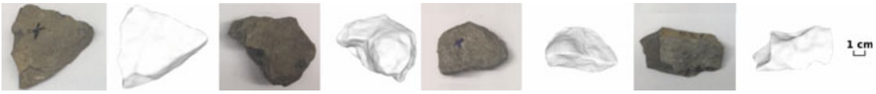
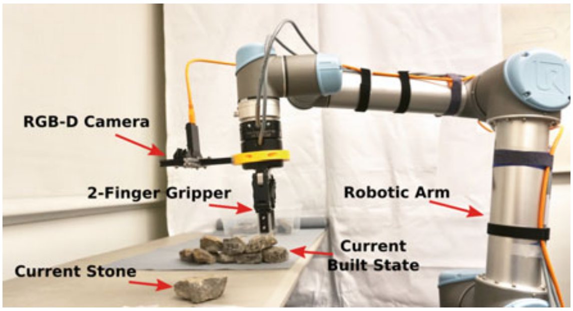


Fig. 3 Irregular shale stones and their corresponding 3D meshes

Fig. 4 An overview of the experimental setup



4.2 Results

The autonomous building system is shown in Fig. 1. In this section, we first compare the proposed algorithm with other work in simulation and physical execution; then, we show the physical execution results.

4.2.1 Stone Tower

Our goal was to build a vertical stone tower using the pre-scanned 3D mesh models as a test structure to evaluate planning under stability constraints. We compared the proposed method with the pose searching algorithm proposed in [6]. The comparisons were conducted in PyBullet physics simulation [25].

The pose searching method used in [6] places each object on the top object of the existing stacking using a physics engine. A cost function is introduced to evaluate the "goodness" of each pose, which considers 4 elements: contact area C_i, kinetic energy E_{kin}, the length between the newly placed object pose P_j and the previously placed object pose P_i (denoted as $\mathrm{r}_{P_j P_i}$), and the dot product between normal of the contact polygon and the trust line direction vector $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$. The cost function is defined as

$$f(P_i) = w_1 C_i^{-1} + w_2 E_{kin}(P_i) + w_3 \|\mathrm{r}_{P_j P_i}\| + w_4 \|\mathrm{n}_i \cdot \mathrm{v}_i\| \,, \tag{3}$$

where w_j are tuned for the object set. After assigning the cost to the valid contact pose, gradient descent is used to search the local optimal pose P_i^*.

Since we use a different type of stone from that of [6], and the size of the stones are also different given that we use different arms and grippers, the w_j given by [6] are not optimal for our application. We also modify the last component of Eq. 3 as

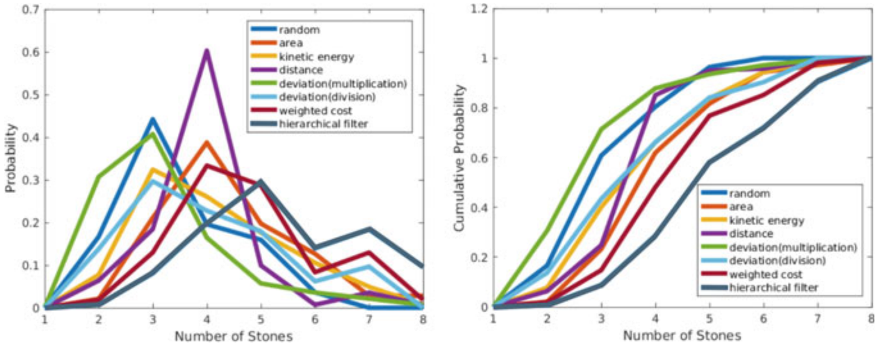


Fig. 5 Vertical tower building results. For each method, we build 150 different vertical stone towers in simulation. The x-axis shows the number of stones each tower has, and on the left figure the y-axis shows the percentage of each height; on the right figure, the y-axis represents cumulative percentage of each height. "random" means that we randomly pick a pose from feasible pose set; "area" represents that the cost function only contains contact polygon area (C) one element, so does "kinetic energy" (E_{kin}), "distance" ($\mathrm{r}_{P_j P_i}$), and "deviation" ($\|\mathrm{n}_i \cdot \mathrm{v}_i\|$). For "deviation", we test both multiplication $\|\mathrm{n}_i \cdot \mathrm{v}_i\|$ and division $\|\mathrm{n}_i \cdot \mathrm{v}_i\|^{-1}$. The "weighted cost" uses the optimized cost function. The proposed "hierarchical filter" significantly outperforms the other methods

$w_4\|\mathrm{n}_i \cdot \mathrm{v}_i\|^{-1}$ given the fact that the larger the dot product is, the smaller the cost should be. At last, we apply Bayesian optimization for Gaussian process modeling called GPyOpt [26] to optimize the weights.

Figure 5 depicts the results of using different cost functions and the proposed hierarchical filter-based algorithm (Algorithm 1). Since the heuristics used in building a vertical tower is different from that of a wall, we modify the filters to fit the task. The filter contains the contact polygon area C, distance $\mathrm{r}_{P_j P_i}$, and the top surface slope. We also evaluate each cost component used in Eq. 3 separately. We can see that the proposed hierarchical filter algorithm has a higher opportunity of building a vertical tower with more than 5 stones compared to all other methods. Table 1 gives the average number of stones each algorithm can build. It also shows that the proposed method can build more stones than other methods. We randomly select 9 towers planned by the proposed method from all of the towers that have a height of at least 6 stones for physical execution. Table 2 shows the building process. Since the first three stones can always be placed successfully, we start the table from the 4th stone. We can see that $4/9$ can be built up to 6 stones, and only $1/9$ drops at the 4th stone. Compared to the towers built in work [6], which only builds up to 4 stones with a chance of $2/11$, our method can build taller stone towers both in simulation and in practice. The reasons for the failures in our execution could be object pose detection error, pickup error, an opening gripper moves an object which is already placed, differences between the object 3D mesh and the real stone, inaccuracies in contact modeling, etc. Figure 6a, b illustrate one tower example of the proposed algorithm in the simulation environment and in practice.

Table 1 Comparison on the average tower height built using different filters

Random	Area	Kinetic energy	Distance	Deviation (m)	Deviation (d)	Weighted cost	Proposed
3.4565	4.3944	4.0845	3.9362	3.2	4.0207	4.75	5.4118

Table 2 Physical execution (d represents stone drops, ✓ represents the successful placements, Ø means that the plan does not contain further steps.)

Stone number	Tower 1	Tower 2	Tower 3	Tower 4	Tower 5	Tower 6	Tower 7	Tower 8	Tower 9
4th	✓	✓	d	✓	✓	✓	✓	✓	✓
5th	✓	✓		d	d	✓	✓	✓	✓
6th	✓	✓				✓	d	✓	d
7th	d	d				Ø		d	

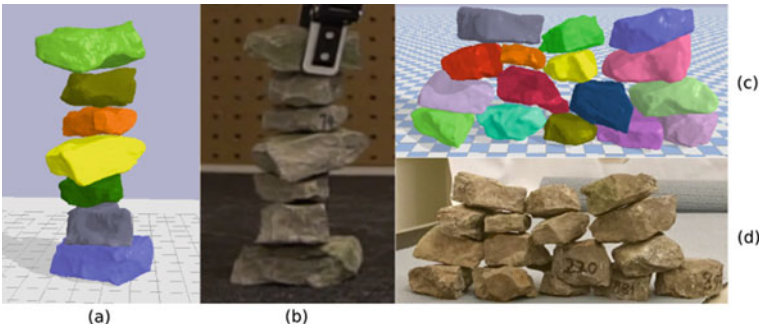


Fig. 6 Vertical tower and wall in simulation and corresponding physical execution results

4.2.2 Stone Wall

In this experiment, the goal is to build a stone wall. The structure is planned in simulation using Algorithm 2, and then the UR5 manipulator places the stones to the planned pose. The planned wall has 4 courses, and each course has 3–5 stones. The execution order is manually calculated, but complies with the assembly order in the simulation if there is no collision during assembly due to the gripper. In other words, the simulated assembly order does not take into account clearances for the fingers or grippers, but if problems exist, reordering stones within a course often fixes potential collisions. As mentioned in the previous section, several things may cause failures during the execution process. We categorize the failure cases into two classes: poor placement and structure collapse. Poor placement contains bad grasping, wrong object pose detection, and stone drops after placement. Structure collapse is the case that after placing the current stone, more than 1 stone falls down. In this experiment, 7 out of 13 walls can be successfully built without collapsing. Table 3 shows the failure

Table 3 Stone wall execution failure rate

	Course 1	Course 2	Course 3	Course 4
Poor placement	0.14	0.18	0.29	0.48
Structure collapse	0	0.04	0.07	0.05

rate during execution. We separate the execution process based on different courses. We can see that as the course increases, the poor placement rate also increases. All the previous minor errors building up to larger errors lead to more failure cases. Figure 6c, d show an example of a planned wall in the simulation and the wall built in the real world using the robotic arm without collapsing. We also compare the proposed algorithm with untrained humans using the same set of stones. We found that humans are much better at executing the generated open-loop plans. However, if the assembly sequences are not provided, our human participants could not plan several steps ahead and retorted to trial-and-error.

5 Conclusion and Future Work

The proposed method is able to plan placements for a set of irregularly shaped rocks and build stable dry-stacked structures. Similar to previous work, we use a rigid body simulation engine in order to find stable poses for rocks. We introduce two primary innovations: first, we use only the physics engine to create a finite set of feasible poses; second, we use a layered refinement architecture that significantly improves performance compared to optimized scalar cost functions in evaluating the quality of feasible poses. We also introduce new filtering terms, which are specific for building walls with interlocking layers, compared to vertical stacks.

We focus on high-level placement planning as it is a central issue in dry stacking. The overall system could be significantly improved with a more specialized and robust execution system, specifically reactively re-planning in the face of errors and unmodeled action outcomes, and in incorporating tactile feedback during placement and pose evaluation. The instructional literature suggests this approach for human builders as well: candidate stones are placed, wiggled, and then either removed or stabilized by wedging small rocks into crevices until the newly placed rock is stable. In any of these situations, having better high-level placement plans that can be executed in an open-loop fashion will be beneficial and this paper represents significant progress in that direction.

Acknowledgements We would like to thank Hironori Yoshida and Dr. Marco Hutter for their valuable input regarding the comparison algorithm, and Jackie Chan for scanning the stones. This work was partially supported by NSF Grant #1846340 and the SMART CoE at UB.

References

1. USEPA: Advancing sustainable materials management: 2014 fact sheet. United States Environmental Protection Agency, Office of Land and Emergency Management, Washington, DC 20460, 22 Nov 2016
2. Green in practice 102—concrete, cement, and CO2
3. Sanders, G.B., Larson, W.E.: Progress made in lunar in situ resource utilization under NASA's exploration technology and development program. J. Aerosp. Eng. **26**(1), 5–17 (2013)
4. Wikipedia Contributors: History of construction—Wikipedia, the free encyclopedia (2019)
5. Chen, I.-M., Burdick, J.W.: Determining task optimal modular robot assembly configurations. In: Proceedings of 1995 IEEE International Conference on Robotics and Automation, vol. 1, pp. 132–137. IEEE (1995)
6. Furrer, F., Wermelinger, M., Yoshida, H., Gramazio, F., Kohler, M., Siegwart, R., Hutter, M.: Autonomous robotic stone stacking with online next best object target pose planning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 2350–2356. IEEE (2017)
7. Helm, V., Ercan, S., Gramazio, F., Kohler, M.: Mobile robotic fabrication on construction sites: Dimrob. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4335–4341. IEEE (2012)
8. Thangavelu, V., Liu, Y., Saboia, M., Napp, N.: Dry stacking for automated construction with irregular objects. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–9. IEEE (2018)
9. Napp, N., Nagpal, R.: Distributed amorphous ramp construction in unstructured environments. In: Distributed Autonomous Robotic Systems, pp. 105–119. Springer (2014)
10. Petersen, K.H., Napp, N., Stuart-Smith, R., Rus, D., Kovac, M.: A review of collective robotic construction. Sci. Robot. **4**(28), eaau8479 (2019)
11. Petersen, K., Nagpal, R., Werfel, J.: An autonomous robotic system for three-dimensional collective construction. Termes, June 2011
12. Lindsey, Q., Mellinger, D., Kumar, V.: Construction with quadrotor teams. Auton Robot **33**(3), 323–336 (2012)
13. Doerfler, K., Sebastian Ernst, Luka Piškorec, Jan Willmann, Volker Helm, Fabio Gramazio, and Matthias Kohler. Remote material deposition. In *International Conference, COAC, ETSAB, ETSAV*, pages 101–107, 2014
14. Saboia, M., Thangavelu, V., Gosrich, W., Napp, N.: Autonomous adaptive modification of unstructured environments. In: Proceedings of Robotics: Science and Systems, vol. XIV (2018)
15. Soleymani, T., Trianni, V., Bonani, M., Mondada, F., Dorigo, M.: Autonomous construction with compliant building material. In: Intelligent Autonomous Systems, vol. 13, pp. 1371–1388. Springer (2016)
16. Saboia, M., Thangavelu, V., Napp, N.: Autonomous multi-material construction with a heterogeneous robot team. In: Distributed Autonomous Robotic Systems. Springer (2018)
17. Liu, Y., Saboia, M., Thangavelu, V., Napp, N.: Approximate stability analysis for drystacked structures. In: 2019 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2019)
18. Liu, Y., Shamsi, S.M., Fang, L., Chen, C., Napp, N.: Deep q-learning for dry stacking irregular objects. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1569–1576. IEEE (2018)
19. Gardner, K.: Stone Building. The Countryman Press (2017)
20. McRaven, C.: Building Stone Walls, vol. 217. Storey Publishing (1999)
21. Vivian, J.: Building Stone Walls. Storey Publishing (1976)
22. Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA) (2011)
23. Zhou, Q.-Y., Park, J., Koltun, V.: Open3D: a modern library for 3D data processing (2018). arXiv:1801.09847

24. Sucan, I.A., Chitta, S.: Moveit! (2013). http://moveit.ros.org
25. Coumans, E., Bai, Y., Hsu, J.: Pybullet physics engine (2018)
26. GPyOpt: Gpyopt: a bayesian optimization framework in python (2016). http://github.com/SheffieldML/GPyOpt

Actively Articulated Wheel-on-Limb Mobility for Traversing Europa Analogue Terrain

William Reid, Gareth Meirion-Griffith, Sisir Karumanchi, Blair Emanuel, Brendan Chamberlain-Simon, Joseph Bowkett, and Michael Garrett

1 Introduction

Exploration of solar system bodies, such as Earth's moon and Mars, has often followed the model of high-coverage remote sensing missions preceding closer inspection via in situ surface operations, first with static landers and successively with mobile assets. In the past decade, Ocean Worlds have garnered substantial interest from the scientific community, precipitating the Europa Clipper mission and Europa Lander study [7]. However, given that the travel time to Europa is significantly longer than to Mars, further measurements of Europa's surface will not become available between the present day and the arrival of the Europa Clipper spacecraft (mid- to late-2020s). This paper presents steps toward a mobile platform design and control, enabling mobility that is inherently robust to a broad range of surface morphologies.

The main contributions of this work are development and field evaluation of an actively articulated control scheme for a wheel-on-limb mobile robot. This approach is agnostic to the mobility mode being used and it scales to highly redundant kinematics. It is kinematically drift-free by construction, and it provides a lightweight computational solution to the real-time inverse kinematics required for active articulation. This technique is evaluated from an energy efficiency standpoint on unstructured Europa analogue terrain.

©2019. California Institute of Technology.

W. Reid (✉) · G. Meirion-Griffith · S. Karumanchi · B. Emanuel · B. Chamberlain-Simon · J. Bowkett · M. Garrett
Jet Propulsion Laboratory, California Institute of Technology, 4800, Oak Grove Drive Pasadena, CA 91109, USA
e-mail: william.reid@jpl.nasa.gov

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_24

1.1 Europa Analogue Surface

Surface mobility on other worlds is challenging for several reasons including, but not limited to terrain uncertainty, autonomy, and risk. Images of Europa taken by Voyager and Galileo have resolutions on the order of hundreds of meters per pixel, with localized regions reaching six meters per pixel. Such resolutions do not provide sufficient knowledge of terrain features at the lander/robot scale to inform the design of terrain-dependent mobility systems.

Terrestrial analogues for surface ice regions on Europa include pentitentes [8], regolith [3], salt evaporites [10], solid ice [6], and chaos terrains [4]. Depending on the modality with which a robotic system is designed to traverse the Europan terrain, each morphology represents a unique set of challenges. Penitentes, which are likely only possible within ±30° of the equator, may not be traversable by any ground-based system, depending on their size and spacing. If not sufficiently small or distributed, line of sight would also likely become an impediment to path planning, autonomous or otherwise. As has been experienced on Mars, regolith presents both wheel slip and embedding challenges to safe surface operations. Both the Mars Exploration Rovers (MERs) and Mars Science Laboratory (MSL) rovers have experienced concerning levels of slip on sandy terrains [1]. A solid ice surface with few or no regions of unconsolidated material may be the least challenging scenario for mobility. If the ice is heavily disrupted, however, as in the case of Europa's well-documented chaos terrains, ground-based travel may become extremely challenging.

Salt evaporites are believed to be formed on Europa when the sublimation of upwelled, brackish material, leaves non-uniform crystalline structures at the surface [10]. Similar morphologies are present in some of Earth's dry lake beds such as the region known as Devil's Golf Course, located in Death Valley, California. Ground mobility on this analogue's salt evaporites is the main subject addressed in this paper; their rough, chaotic, and undulatory nature pose several unique mobility challenges.

1.2 Versatile Mobility

Given the significant terrain uncertainty, it is postulated that a wheeled mobility system that is capable of wheeled driving over benign terrain and combined wheeled and limbed motions over more complex terrain is needed. For the purposes of this work, the high Degree-of-Freedom (DoF) RoboSimian quadruped robot, shown in Fig. 1, is used to evaluate versatile mobility capabilities. RoboSimian belongs to a class of robots known as actively articulated wheel-on-limb platforms that have been used in previous work to traverse unstructured planetary analogue terrain [5, 11, 15, 16]. Compared with conventional planetary exploration robots such as the MSL rover, wheel-on-limb platforms may articulate their high DoF limbs to actively morph their shape to terrain profiles, as well as redistributing center of mass position relative to wheel/ground contact points.

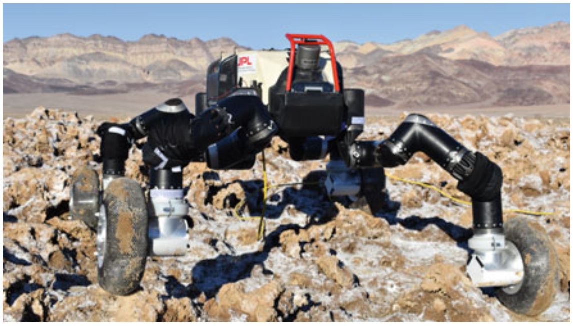


Fig. 1 The RoboSimian wheel-on-limb rover traversing Europa analogue unstructured terrain composed of salt evaporites in Death Valley, California, USA. The RoboSimian rover is capable of conforming its limbs to the terrain while its wheels and limbs propel the platform forward

Four mobility modes are implemented that combine joint articulation in different ways to achieve locomotion. These modes are driving (wheel-rolling), inchworming (push-rolling), sculling, and wheel walking. Driving is the most conventional mode where locomotive forces are sourced only from the wheels being actuated. During inchworming the chassis moves continuously while the front and back limb pairs alternate pushing across the terrain and working as a motion anchor by keeping the limb end-points stationary. In Wettergreen et al. [15], the efficacy of inchworming compared to driving on soft-soil covered slopes is highlighted. Sculling is similar to inchworming where the motion is divided into three sections: front limb pair driving, back limb pair driving, and chassis moving. Lastly, wheel walking moves a single limb at a time, maintaining wheel/ground contact during this limb motion. After all limbs have moved, the body is moved while keeping the limbs stationary. Wheel walking was posed as a solution to traversing unstructured terrain for the JPL ATHLETE rover in [13]. A contribution of our work is the comparison of these four mobility modes on Europa analogue terrain using an energy efficiency metric.

In addition to these modes, an actively articulated suspension control scheme is needed for reliable locomotion over unstructured terrain. This is demonstrated in field trials performed with the ATHLETE rover in [14] and the Sherpa_TT rover in [5]. In both, articulated suspension acts to maintain traction at each wheel using force feedback at the wheels while also maintaining body attitude via inertial measurement unit feedback. Such a control architecture is used in this paper, however, extended to multiple mobility modes and a robot with redundant limbs. To extend to seven DoF redundant limbs, the controller uses a pre-computed lookup-table to calculate its inverse kinematics. Such an inverse kinematics solver has been used for controlling the RoboSimian platform in [12] so as to ensure computationally efficient generation of safe paths through the robot's 32 DoF joint space. The computational complexity of the proposed architecture is minimal and can generalize to radiation-hardened computers. These limited computational capacity computers are a bottleneck for traditional robotic methods as highlighted in [2].

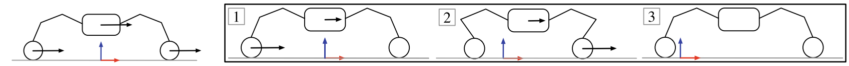
The RoboSimian robot was originally designed as a mobile manipulation platform for the DARPA Robotics Challenge [9]. This quadruped has since been adapted to include four additional driven wheels making it a wheel-on-limb robot with 32 independently actuated rotary joints, with each joint having an electromechanical brake. The total mass of the platform is 120 kg. It is noted that the robots have been adapted and not explicitly designed for the study described in this paper. The absolute energy efficiency values reported from traverses are considerable given the power required to hold and drive all RoboSimian limb actuators. A topic of future work is to use the energy efficiency comparisons presented in this work to inform design of a future rover mobility system. This system would be designed to utilize mobility modes that have been highlighted as both effective in traversing terrain and energetically efficient.

2 Mobility Modes

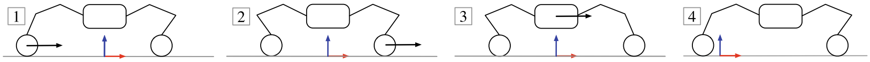
The four mobility modes used in this work include driving, inchworming, sculling, and wheel walking. Figure 2 presents the distinct gait sequences of each mobility mode using wireframe sequence diagrams of a generic internally articulated rover traversing flat terrain.

In Fig. 2a, driving propels the rover along its direction of travel by only relying on the wheel actuators. Unlike driving, the remaining mobility modes leverage the other limb actuators in addition to the wheels for forward progression.

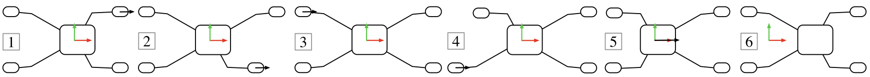
The inchworming mode anchors two contact points while moving the rest of the robot forward. The inchworming gait alternates anchored and driving contact points causing the limbs to compress toward each other and extend away from each other. As shown in Fig. 2b, in frame 1, inchworming starts with the front wheels stationary and the back wheels driving forward. All limb joints must move to accommodate for these constraints causing the central chassis of the rover's body to move at half the rate of the back wheels. In frame 2, the rover is in a compressed limb state and



(a) Driving (b) Inchworming



(c) Sculling



(d) Wheel Walking

Fig. 2 Wireframe sequence diagrams of each of the four mobility modes used to traverse Europa analogue terrain. Driving (**a**), inchworming (**b**), and sculling (**c**) are presented from the side as these modes are symmetric about the direction of travel. Wheel walking (**d**) is presented from a top-down view to show how individual limbs move at different stages of a gait. In any frame the x-axis is red, the y-axis is green, and the z-axis is blue

alternates its stationary and drive wheels so that the back wheels are still and the front wheels begin to move. Again, all limbs must move to comply with the contact point motion. In frame 3, the rover is once again in its extended limb state and is ready to restart the gait cycle.

Sculling is a variant of inchworming where a pair of limbs move while the other two limbs and chassis remain stationary. In Fig. 2c, in frame 1, the rover starts in a state where the back limbs are extended away from the chassis and the front limbs are compressed toward the chassis. Motion begins with the back limbs driving forward, while the chassis and front limbs stay stationary. In frame 2, the rover is in a compressed state and the front wheels begin to move leaving the chassis and back wheels stationary. In frame 3, the final motion of the gait propels the body forward while all contact points remain stationary. The rover returns to its starting state in frame 4, now ready to restart the sculling gait.

Sculling may be modified in a variety of ways. Firstly, the sets of limbs do not have to be divided into the forward and back limbs. Sets of diagonal limbs or sets of side limbs may be moved or held stationary together. Additionally, sculling may be modified so that the chassis is lowered into contact with the terrain while the chassis is not moving. Having the chassis in contact with the ground creates an additional anchor point resulting in additional friction for the moving wheels to propel against. This modification may be particularly useful on icy or sandy slopes where slip is a significant factor affecting forward progression. This is a modification to be considered in future work, where mobility on alternative Europan analogues such as soft regolith and inclines covered in solid ice is to be investigated.

Wheel walking is a further instantiation of inchworming where only one wheel is moved at time. As shown in the sequence in Fig. 2d, the first four frames show individual limbs compressing toward or away from the rover chassis. In frame 5, the chassis is moved forward while the contact points are held stationary. Compared to conventional statically-stable quadrupedal walking, this mode does not lift any wheel off the surface. Therefore, the center of mass does not need to be moved to comply with stability polygon contstraints between limb motions. Such a wheel walking strategy was demonstrated with the ATHLETE rover [13] as a faster alternative to a limb-lift walking gait.

3 Actively Articulated Suspension

We now present an actively articulated suspension controller that works with all of the presented mobility modes. The first objective of the controller is to ensure that there is a tractive force at each wheel and therefore propulsion of the platform is distributed to all wheels. The second objective is to maintain chassis roll and pitch relative to the robot-centric gravity-aligned $\{RCGA\}$ frame labeled in Fig. 3. This objective ensures that extra-proprioceptive sensors such as stereo-camera pairs are not exceedingly pitched or rolled enabling a reliable visual odometry localization

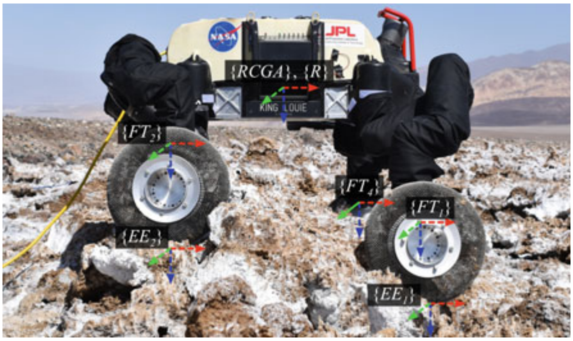


Fig. 3 Actively articulated suspension requires limb i's end-point situated at the end effector frame $\{EE_i\}$ to move relative to a robot-centered gravity-aligned $\{RCGA\}$ frame. Six DoF force/torque sensors are located at the interface between a limb and its wheel, centered on the force-torque frame $\{FT_i\}$. The robot frame $\{R\}$ is fixed to the robot chassis

solution. The controller's output is a desired trajectory for each limb end-point, located at frame $\{EE_i\}$ for limb i, also labeled in Fig. 3. Each limb end-point is constrained to moving on a plane that is parallel to the xz^{RCGA} plane.

3.1 Driftless Motion for Redundant Limbs

Given that force control is being used to actively articulate RoboSimian's limbs, a high controller rate ($>$100 Hz) is required. A common problem for high rate optimization and Jacobian pseudo-inverse-based motion control of redundant manipulator arms is *kinematic drift*, where limb joints may drift within the arm's null-space over repeated motions. A RoboSimian limb i's workspace $\mathscr{W}_i$ is the space the $\{EE_i\}$ frame may move in relative to the robot's chassis frame $\{R\}$. To execute desired motions in $\mathscr{W}_i$, we must ensure that motions through the limb's joint (configuration) space $\mathscr{C}_i$ are repeatable and have no collisions. One way of accomplishing this is to pre-generate collision-free $\mathscr{C}_i$ motions that have a one-to-one mapping, f, with motion in $\mathscr{W}_i$.

The mapping f is generated through a Breadth-First-Search (BFS) expansion through grids that span each limb's workspace. The search over a single limb's workspace begins at a pre-defined limb end-point with a known limb joint configuration. Using a Jacobian-based velocity kinematic model of the limb, each BFS expansion drives the limb from a known configuration to a new configuration by a workspace grid cell size. An expansion is deemed valid if there are no collisions between limbs or between the limb and the rover chassis. Once all grid cells have been visited by the BFS expansion routine the formulation of the mapping f is complete taking the form of a two-dimensional lookup-table. This lookup-table is interpolated over to associate a coordinate in $\mathscr{W}_i$ with a coordinate in $\mathscr{C}_i$.

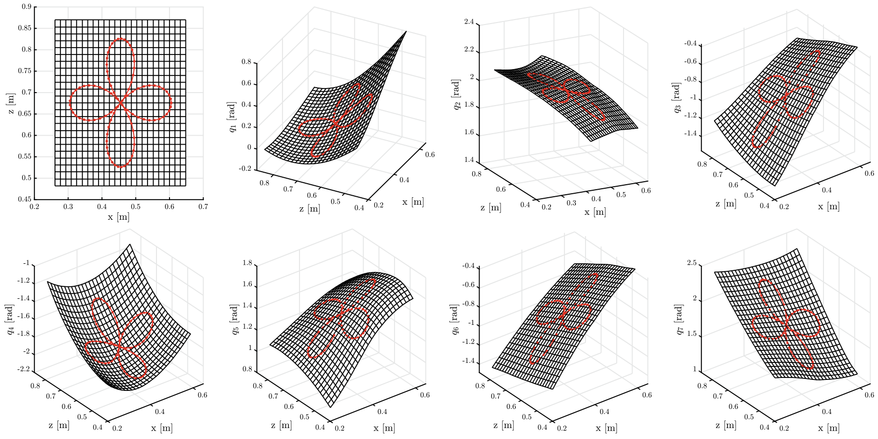


Fig. 4 The mapping between RoboSimian's limb 1 workspace $\mathscr{W}_1$ to the limb 1 joint (configuration) space $\mathscr{C}_1$. The remaining figures show the associated joint manifolds for the first seven joints along limb 1. A red limb end-point rose path is generated in $\mathscr{W}_1$ and then interpolated into $\mathscr{C}_1$. Joint 8 is not shown as it does not affect the position of the limb end-point. $\mathscr{W}_1$ is relative to $\{R\}$

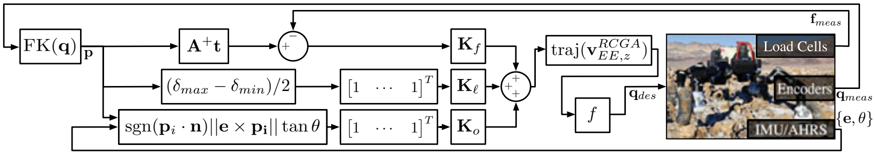

$\text{FK}(\mathbf{q})$
$\mathbf{p}$
$\mathbf{A}^+\mathbf{t}$
$+$
$-$
$\mathbf{K}_f$
$(\delta_{max} - \delta_{min})/2$
$[1 \ \cdots \ 1]^T$
$\mathbf{K}_\ell$
$\text{sgn}(\mathbf{p}_i \cdot \mathbf{n})\|\mathbf{e} \times \mathbf{p_i}\| \tan\theta$
$[1 \ \cdots \ 1]^T$
$\mathbf{K}_o$
$+$
$+$
$+$
$\text{traj}(\mathbf{v}^{RCGA}_{EE,z})$
f
$\mathbf{q}_{des}$
Load Cells
Encoders
IMU/AHRS
$\mathbf{f}_{meas}$
$\mathbf{q}_{meas}$
$\{\mathbf{e}, \theta\}$

Fig. 5 The actively articulated suspension controller

Figure 4 presents a grid in $\mathscr{W}_1$ that has a mapping to $\mathscr{C}_1$. The association between each workspace coordinate and each joint space coordinate is shown by each of the seven joint to work space surfaces in Fig. 4. Also in Fig. 4, an illustrative rose limb end-point path is plotted in $\mathscr{W}_i$ and is interpolated on each of the joint surfaces to formulate an associated joint space path.

3.2 Feedback Controller

To retain wheel traction, regulate body orientation and keep limbs away from suspension saturation while traversing undefined terrain topologies the limb end-points can raise and lower relative to the chassis. These four raise/lower commands are the output of a multi-objective feedback controller summarized in Fig. 5 and inspired by the controller described in [5].

The overall actively articulated suspension control policy is the summation of each of the three controller errors mulitiplied by proportional gains:

$$\mathbf{v}_{EE_i,z}^{RCGA} = K_f(\mathbf{f}_{ref}^{RCGA} - \mathbf{f}_{meas}^{RCGA}) + K_o(\mathbf{o}^{RP}) + K_\ell(\ell_{err}), \tag{1}$$

where $\mathbf{f}_{ref}^{RCGA}$ is a vector of the expected vertical limb end-point forces for a static system, $\mathbf{f}_{meas}^{RCGA}$ is a vector of the vertical load cell force measurements projected into $\{RCGA\}$, $\mathbf{o}^{RP}$ is a vector of the limb end-point vertical deflections required to keep the roll and pitch at their desired angles, and ℓ_{err} is the amount all limb end-points have to move to maximize distance away from suspension saturation across all limbs. These velocities are then transformed into $\{R\}$, and then into trajectories of desired points in each limb workspace with associated timestamps. Each of these points in $\mathscr{W}_i$ are then mapped to $\mathscr{C}_i$. These joint trajectories are then executed. A block diagram summarizing the overall controller architecture is shown in Fig. 5.

The controller is composed of three separate components: a force controller, an orientation controller, and a leveling controller. In the force controller, error is calculated as the difference between $\mathbf{f}_{ref,z}^{RCGA}$ and $\mathbf{f}_{meas,z}^{RCGA}$. A load cell at frame $\{FT_i\}$, shown in Fig. 3, reports a three DoF force measurement $\mathbf{f}_{i,meas}^{FT}$. This force is transformed into $\{RCGA\}$, and the vertical component for each limb is used to assemble the 4×1 vector $\mathbf{f}_{meas,z}^{RCGA}$. Reference forces are calculated using the system of equations that describe the static moments and forces assuming all four wheels are in contact with the ground. The system of equations is $\mathbf{A}\mathbf{f}_{ref}^{RCGA} = \mathbf{t}$, where A is a 3×4 mapping matrix and t is a 3×1 solution vector equal to $\begin{bmatrix}0 & 0 & mg\end{bmatrix}^T$. The first two equations describe the moments about the center of mass, while the third equation describes the forces along the $\{RCGA\}$ vertical axis. This redundant system of equations may be solved using the Moore–Penrose pseudo-inverse: $\mathbf{f}_{ref}^{RCGA} = \mathbf{A}^+\mathbf{t}$.

The objective of the orientation controller is to produce a vertical offset error value for each limb end-point. It is desired for the xy^R plane of the rover chassis to align with the xy^{RCGA} plane of the $\{RCGA\}$ frame. The limb end-points are projected onto the xy^R plane giving a contact point position vector $\mathbf{p}_{xy,i} = \begin{bmatrix}p_{x,i} & p_{y,i} & 0\end{bmatrix}$ for each limb. The vertical position offsets o_i^{RP} that represents the distance between the xy^R plane and the desired xy^{RCGA} plane are then calculated. The difference in orientation is expressed in axis-angle form as $\{\mathbf{e}_{err}, \theta_{err}\}$. Given that only roll and pitch rotation errors are considered, $\mathbf{e}_{err}$ is always in the xy^{RCGA} plane. The vertical offset of $\mathbf{p}_{xy,i}$ to the xy^{RCGA} plane is d_i, and is expressed as $d_i = ||\mathbf{e}_{err} \times \mathbf{p}_{xy,i}||$. The magnitude of the vertical offset distance may then be expressed as $|o_i^{RP}| = d_i \tan(\theta_{err})$. The sign of the vertical offset may be found by $\operatorname{sgn}\left(\mathbf{p}_{xy,i} \cdot (\mathbf{e}_{err} \times \mathbf{k})\right)$, where $\mathbf{k}$ is the unit vector along the $\{RCGA\}$ z-axis, $\mathbf{k} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. The expression for o_i^{RP} is therefore: $o_i^{RP} = \operatorname{sgn}\left(\mathbf{p}_{xy,i} \cdot (\mathbf{e}_{err} \times \mathbf{k})\right) d_i \tan(\theta_{err})$.

The leveling controller acts to introduce a bias that drives all limb contact points up or down away from the workspace limits. Figure 6 shows the geometry of each of the parameters used in the leveling controller. To begin with, the limb end-point distance that is closest to the maximum workspace boundary, δ_{max}, is found, followed by the limb end-point distance that is closest to the minimum workspace boundary δ_{min}. The limb controller error is found as the shared distance away from workspace limits: $\ell_{err} = (\delta_{max} - \delta_{min})/2$.

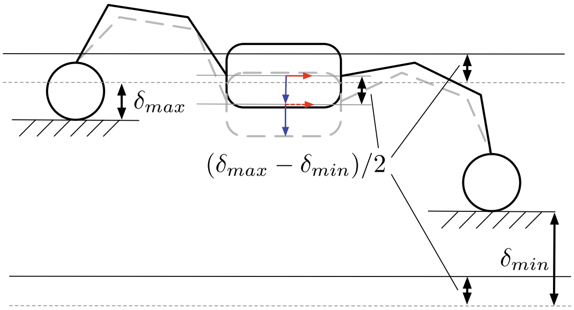


Fig. 6 The leveling controller acts to offset al.l limb end-points from workspace limits an equal amount $\ell_{err} = (\delta_{max} - \delta_{min})/2$. The initial rover configuration is shown by gray dotted lines, while the desired rover configuration with the leveling controller offset applied to each limb is shown with black solid lines

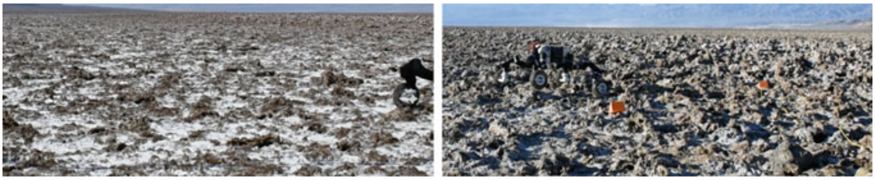
4 Experiment Design

To validate each of the aforementioned mobility modes and the mobility mode-agnostic actively articulated suspension controller for use on Europan analogue terrain, a series of traverses were conducted with the RoboSimian platform at a dry lake bed with salt-evaporite formations in Death Valley, CA, USA. A total of four 10 m long traversal sites were chosen with increasing levels of terrain roughness. Figure 7 shows terrain from each of the four test sites. All traversals performed had the robot drive either straight forward or straight backwards, each using one of the four mobility modes with actively articulated suspension enabled. Table 1 summarizes each of the trials performed and the total distance driven for each mobility mode at a specific site. A traverse was deemed successful if the robot moved greater than 1 m into a traverse. Termination conditions for a trial included a robot limb getting stuck on terrain, a technical fault preventing further traversal, or the operator issuing a stop command when the end of the traverse site was reached.

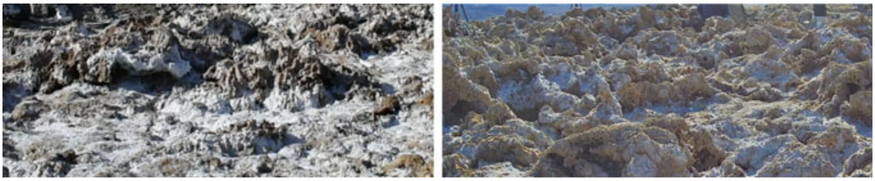
Ground truth traverse distances reported in Table 1 were measured with a tape measure after a trial was completed. Traverse velocity was measured by visual odometry that used a stereo-camera pair positioned at the front of the robot. Energy usage was measured using current and voltage sensing on each limb. All current, encoder values, and force/torque data were measured at a frequency 250 Hz. Video footage of each traverse was collected.

5 Experiment Results

Statistics describing each trial are summarized in Table 1. In total, 27 trials were run on the four sites, with a total traverse distance of 179 m.



(a) Site 1 (b) Site 2



(c) Site 3 (d) Site 4

Fig. 7 The four sites that were traversed by the RoboSimian platform using different mobility modes. The sites were selected to be progressively rougher in numeric order

Table 1 Summary of mobility mode performance on each of the four terrains. The data presented include the number of trials for each site/mode pair, the average speed reported by visual odometry, the average traverse distance, the average energy efficiency, the energy efficiency of the chassis moving component of sculling and walking, the fraction of time spent moving the chassis, the average power usage, and the standard deviation of the power usage. Additionally, controller error magnitudes for limb forces, chassis roll/pitch, and ℓ_{err} are presented

	Site	Num. Trials	Avg. V.O. Speed [cm/s]	Dist. [m]	Avg. Dist./Trial [m]	Avg. Energy Eff. [cm/kJ]	Mov. Eff. [cm/kJ]	Mov. Frac.	Avg. Pow. [kW]	Pow. Std. Deviation [W]	Avg. f_{err} [N]	Avg. pitch err. [°]	Avg. roll err. [°]	Avg. ℓ_{err} [cm]
Drive	1	5	5.8	51.4	10.3	7.4	-	-	0.79	225	20.8	0.6	0.4	3.4
	2	2	5.8	13.0	6.5	6.1	-	-	1.06	267	32.4	1.0	0.5	3.5
	3	2	5.7	8.8	4.4	6.4	-	-	0.96	245	33.9	1.0	0.5	3.2
	4	1	4.4	3.4	3.4	5.1	-	-	0.88	173	33.1	0.8	0.7	8.6
Inch.	1	4	3.8	37.3	9.3	4.3	-	-	0.97	328	23.0	1.0	0.3	4.6
	3	2	3.2	11.4	5.7	4.1	-	-	0.89	519	31.7	1.3	0.6	4.6
	4	3	3.7	7.8	2.6	3.7	-	-	1.14	380	29.8	1.5	0.5	5.4
Scull	1	2	1.9	18.6	9.3	2.9	8.7	0.26	0.79	307	21.4	0.6	0.3	6.7
	3	1	2.0	2.7	2.7	2.8	9.4	0.20	0.93	484	25.4	0.9	0.4	6.1
Walk	1	2	1.3	13.4	6.7	1.6	7.4	0.16	0.84	370	31.7	0.8	0.3	12.0
	3	2	1.5	8.5	4.2	2.1	9.3	0.15	0.99	336	29.6	0.9	0.6	6.3
	4	1	1.7	2.8	2.8	2.1	9.7	0.14	1.05	379	23.6	0.9	0.4	4.9

It is observed that modes operate at different speeds with driving being the fastest, followed by inchworming, then sculling and lastly wheel walking. A body motion of 5 cm/s was the operator-defined speed for driving, sculling, and wheel walking. Sculling and walking progressed at a slower rate than driving and inchworming due to the components of their gaits where the chassis stayed stationary. Discrepancies in the average speed values presented in Table 1 and the desired rates may be due to noise on the visual odometry reading.

The rate at which traverses are completed is reflected in the energy efficiency of the platform for each mode. Speed and energy efficiency are correlated, with the faster modes performing more efficiently than the slower modes. The energy efficiency is calculated as the product of the velocity and the robot joint power consumption: $\eta = ||\mathbf{v}_R^{RCGA}||/(\Sigma_{j=1}^{n=32} V_j I_j) \times 10^4$, where η is the energy efficiency in cm/kJ, j is the robot joint index, RoboSimian has $n = 32$ actuated joints, V_j is joint j's voltage, I_j is joint j's current consumption, and $\mathbf{v}_R^{RCGA}$ is the chassis's velocity relative to $\{RCGA\}$.

The energy efficiency of each mode at various terrain sites is presented in Fig. 8. Each of the subplots in this figure show a normalized distribution of all energy efficiency samples recorded from each trial associated with an individual terrain site and mobility mode pair. Energy efficiency data was sampled 250 Hz. These distributions show that driving and inchworming have a single peak given that the body is in constant motion throughout a traverse. It is noted that for driving on terrain 4 and all inchworming trials there are a large number of samples close to zero due to brief pauses between the end of one gait and the beginning of another. For sculling and wheel walking two peaks are present, one dominant peak near 0 cm/kJ for when the limbs are moving and another near the average energy efficiency measure for chassis motion. A k-means clustering with two clusters was performed for both the sculling and wheel walking distributions. The resulting average energy efficiency for the chassis motion phase along with the fraction of time that the chassis is in motion is presented in Table 1. In an ideal case, a sculling gait moves the chassis 33% of the time. For wheel walking, the chassis should move for 20% of the time. The lower motion fraction numbers presented in Table 1 are caused by brief motion pauses between gait cycles.

The power consumption generally increases with terrain roughness. Much of the power consumption is caused by the continuous holding torque (roughly 750 W) required to keep the platform suspended and not collapse under gravity. This holding torque is required while actively articulated suspension is activated and each of the joints' electromechanical brakes are not engaged. This power usage is a significant proportion of the overall power usage required during a traverse. The standard deviation on power usage is larger for modes with varying wheel footprints. For inchworming, sculling, and wheel walking the limbs have to reach out a long distance and also compress toward the center of the chassis. The large moment arms during the extension component of the gait results in larger holding torques on joints, while the smaller moment arms have the opposite smaller torque effect in the compressed state.

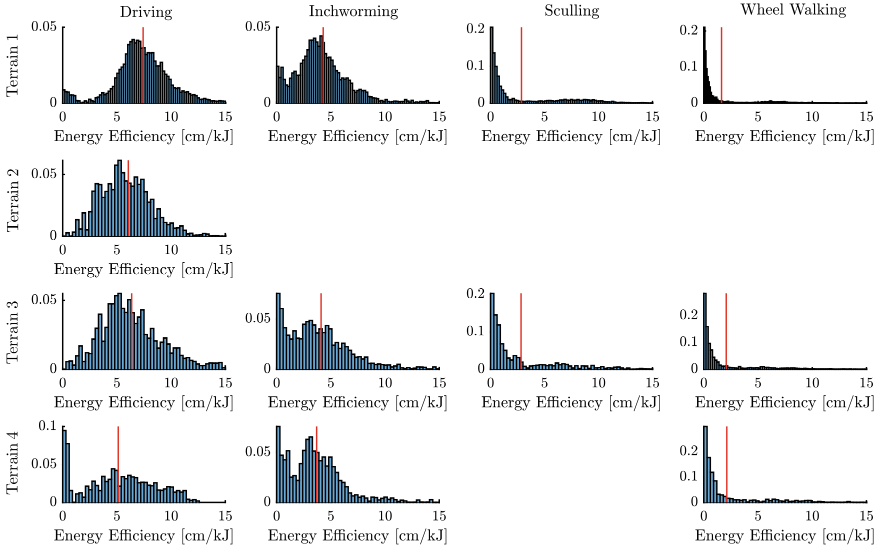


Fig. 8 Energy efficiency sample distributions for each mobility mode at the different sites. The red line on each histogram shows the position of the mean average energy efficiency. The y-axis on each histogram is the normalized quantity of samples. The sculling and walking modes are bi-modal: they have a limb motion phase and a chassis motion phase. In each of the sculling and walking histograms, there are peaks around 0 cm/kJ and another much smaller peak at the energy efficiency average for the motion phase of the mode

Another objective of this paper is to validate the actively articulated suspension controller. Average error magnitudes for mobility mode and terrain pairs are presented in Table 1. Average forces are shown to be below 35 N across all trials, with all four wheels in contact with the terrain across all trials. Roll and pitch averages are below 2° for all trials, which is satisfactory for sensor pointing for visual odometry. In regard to the leveling controller, at no point in any trial did any of the limbs hit their suspension limits. Figure 9 shows the force and orientation controller performance during two illustrative trials: driving on terrain 1 and wheel walking on terrain 4. In both trials, wheel 2 is continuously in contact with the ground and the measured force stays within a 50 N range of the reference force. For both roll and pitch plots, there is no more than a 2.5° deflection about the roll or pitch axis.

Qualitatively, the design of the robot's wheel hub resulted in many "snagging" events in which the wheel hub caught the terrain, thus preventing the robot from continuing its traverse. An image of such a hub snag event is shown in Fig. 10. To prevent this in the future, the actively articulated suspension controller may be modified to use force feedback data projected along the x-axis of $\{RCGA\}$. Another common problem encountered was internal stress buildup within the robot's kinematic structure. This was caused by wedging effects in which the wheels would be compressed toward or pushed away from the chassis by irregular terrain. Like

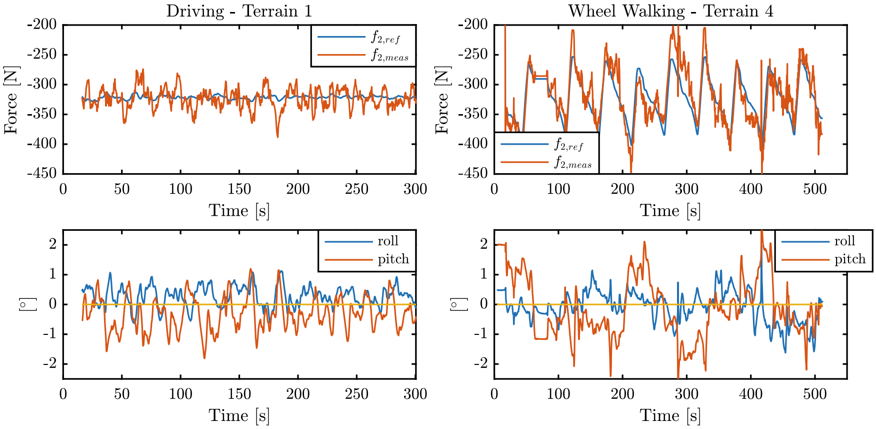
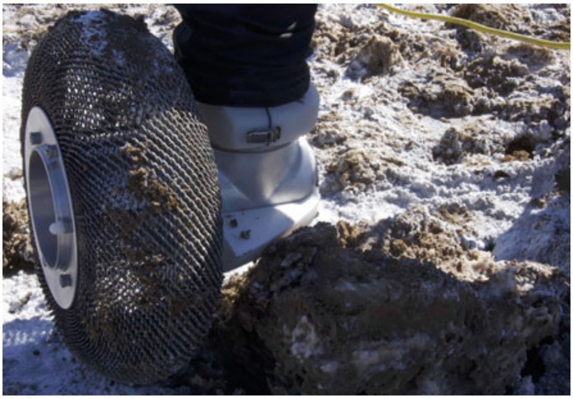


Fig. 9 Illustrative plots showing how the actively articulated suspension controller acts during a driving trial and a wheel walking trial

Fig. 10 A common problem during traverse over salt evaporites was the RoboSimian wheel hubs snagging on terrain protrusions. This problem may be remedied by modifying the hub design in addition to including force feedback in the actively articulated suspension controller along the x^{RCGA}-axis as well as along y^{RCGA}-axis



the snagging effects, this problem may be addressed by including additional force feedback using forces projected along the y^{RCGA}-axis.

6 Conclusions and Future Work

This work is motivated by the need for a highly adaptable mobility system for the in situ exploration of unstructured Europa analogue terrain. Four distinct mobility modes in combination with an actively articulated suspension controller are presented and evaluated at a Europa analogue test site. Over varied terrain roughness, the efficacy of actively articulated suspension in coordination with all mobility modes is demonstrated. Energy efficiency results on all terrains show that the driving mode is most efficient followed closely by inchworming. Sculling and wheel walking are generally more than twice as expensive than driving.

Even though the sculling and wheel walking modes were shown to be relatively inefficient, the use of modified gaits that place the belly of the robot on the ground during limb motions is promising. By allowing the chassis of the robot to take a portion of the total system weight, the joint actuators would spend less energy fighting gravity just to suspend the system.

Problems that impeded locomotion for all modes included wheel hubs snagging on terrain protrusions and wedging effects on wheels that resulted in build up of internal stresses. In future work, the actively articulated suspension controller will be modified so that limbs may comply not only along axes that are parallel to the robot-centric gravity-aligned z-axis. Further evaluation of the various mobility modes on sloped terrain is also desirable.

Acknowledgements We would like to acknowledge the National Parks rangers at Death Valley for their guidance in operating in the Death Valley National Park. The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. ©2019. California Institute of Technology. Government sponsorship acknowledged.

References

1. Arvidson, R.E., Iagnemma, K.D., Maimone, M., Fraeman, A.A., et al.: Mars science laboratory curiosity rover megaripple crossings up to sol 710 in Gale Crater. J. Field Rob. **34**(3) (2017)
2. Bajracharya, M., Maimone, M., Helmick, D.: Autonomy for mars rovers: past, present and future. Computer (2008)
3. Buratti, B., Golombek, M.: Geologic implications of spectrophotometric measurements of Europa. Icarus **75**(3) (1988)
4. Collins, G., Nimmo, F.: Europa, chap. Chaotic Terrain on Europa, pp. 259–281. The University of Arizona Press (2009)
5. Cordes, F., Kirchner, F., Babu, A.: Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain. J. Field Rob. (2018)
6. Greeley, R., Sullivan, R., Klemaszewski, J., Homan, K., et al.: Europa: initial Galileo geological observations. Icarus **135**(1) (1998)
7. Hand, K.P., et al.: Europa lander study. Tech. Rep. JPL D-97667, NASA JPL (2016)
8. Hobley, D.E.J., Moore, J.M., Howard, A.D., Umurhan, O.M.: Formation of metre-scale bladed roughness on Europa's surface by ablation of ice. Nat. Geosci. **11**(12) (2018)
9. Karumanchi, S., Edelberg, K., Baldwin, I., Nash, J., et al.: Team roboSimian: semi-autonomous mobile manipulation at the 2015 DARPA robotics challenge finals. J. Field. Rob. **34**(2) (2016)
10. McCord, T.B., Hansen, G.B., Fanale, F.P., Carlsen, R.W., et al.: Salts on Europa's surface detected by Galileo's near infrared mapping spectrometer. Science (1998)
11. Reid, W., Pérez-Grau, F.J., Göktoğan, A.H., Sukkarieh, S.: Actively articulated suspension for a wheel-on-leg rover operating on a Martian analog surface. In: IEEE ICRA (2016)
12. Satzinger, B.W., Lau, C., Byl, M., Byl, K.: Tractable locomotion planning for RoboSimian. Int. J. Rob. Res. **34**(13) (2015)
13. Townsend, J., Biesiadecki, J.: Sliding gait algorithm for the all-terrain hex-limbed-terrestrial explorer (ATHLETE). In: ASME DSCC (2012)
14. Townsend, J., Biesiadecki, J., Collins, C.: ATHLETE mobility performance with active terrain compliance. In: IEEE AeroConf (2010)

15. Wettergreen, D., Moreland, S., Skonieczny, K., Jonak, D., et al.: Design and field experimentation of a prototype lunar prospector. Int. J. Rob. Res. **29**(12) (2010)
16. Wilcox, B.H., Litwin, T., Biesiadecki, J., Matthews, J., et al.: ATHLETE: A cargo handling and manipulation robot for the Moon. J. Field Rob. **24**(5) (2007)

Meander-Based River Coverage by an Autonomous Surface Vehicle

Nare Karapetyan, Jason Moulton, and Ioannis Rekleitis

1 Introduction

Significant work has been done in the field of autonomous area coverage [3, 6], utilizing single or multiple robot systems. Depending on the goal or constraints of the coverage operation different variations have been presented such as coverage under limited resources [12, 20], information-driven coverage [13], complete coverage [7, 19], sampling-based coverage and passive coverage [11]. One of the most common approaches is known as boustrophedon coverage, that performs back and forth or "lawn mower" motions [1, 2]. This method performs very well in open areas that do not have many changes in the shape [12, 19]. Coverage is a central problem in multiple domains, such as agriculture, environmental monitoring, search and rescue, and marine exploration. For each domain, the desired solution has different priorities, and can be influenced by different factors. Therefore, having prior knowledge about the physics of the environment can enhance coverage strategies applied in that domain (Fig. 1).

To the best of our knowledge, we were the first to implement systematic coverage in a riverine environment [8]. These algorithms differ from well-known approaches in a way that they provide domain-specific geometric solutions. For the tight and uneven spaces, prior algorithms were not as effective. The following work addresses the river coverage problem, using current flow speed information conditioned by the meanders of the river, to enhance time and energy efficiency of the robot's operation.

N. Karapetyan (✉) · J. Moulton · I. Rekleitis
University of South Carolina, Columbia, SC, USA
e-mail: nare@email.sc.edu

J. Moulton
e-mail: moulton@email.sc.edu

I. Rekleitis
e-mail: yiannisr@cse.sc.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_25

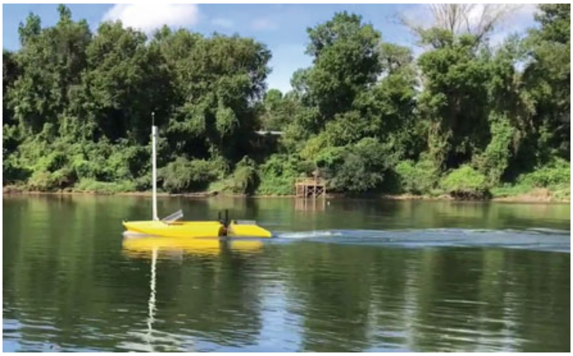


Fig. 1 Autonomous Surface Vehicle (ASV) during coverage operations at the Congaree River

River morphology is quite complicated, especially, the twists and turns a river takes, called meanders. Due to these curves, the water flows at different speeds across the river, which in turns contributes to different rates of erosion and sediment deposits. ASV operations are heavily affected by the water currents encountered [15], as such, the efficiency of the ASV's navigation can be improved by choosing to go against the slower currents and with the faster currents.

In the following section, we present a survey of related work. Next, Sect. 3 will define the coverage problem and present the meander based algorithm with some discussion on improvement of that method as well. Section 4 presents the experimental setting and the results of field trials with discussion of outcomes. Finally, Sect. 5 gives a summary of the proposed method and some remarks on the future work.

2 Related Work

Substantial work has been done on the design and operations of Autonomous Surface Vehicles (ASVs) in rivers. One of the works show how to design and operate ASVs for performing bathymetric surveys [5]. There has been some work also studying the problem of navigating a river with an ASV [21].

In our recent work on river coverage [8], we have proposed different coverage techniques that vary by the available resources and the application domain. One of the approaches proposed was termed zig-zag (Z-cover) approach performing a single pass over a river segment by bouncing back and forth across the shores of the river. Such an approach is utilized when there is limited time available (e.g. fuel constraints). Another approach termed Transversal Coverage (T-cover) performed boustrophedon coverage in between the shores. While inefficient due to the many turns, T-cover is utilized during sediment studies. Finally, the most efficient was Longitudinal coverage (L-cover), which is a complete coverage algorithm. It decomposes the river along the length based on the width information, such that each cluster of the river has approximately the same width. Then each of the clusters is decomposed into passes based on the size of the sensor footprint of the vehicle. Nevertheless, in that work, we do not consider any of the geological properties of rivers. As such some work has been done to find connection between river meanders and the speed of river current [16, 17] or to model the external forces [15]. As a matter of fact the first one that understood how flow affects the length of meanders and its down-flow migration was Albert Einstein [4]. In another work, Kai Qin and Dylan A. Shell use the well-studied model of the geometry of meanders to estimate the shape of the unseen portion of the river [17]. Using this information as an input, a boat can adjust the speed and perform more optimal and smooth paths when performing online navigation.

The problem approached in this paper is a variation of the well-studied coverage problem [6]. Of particular relevance are two works dealing with the coverage of rivers using drifters: vehicles that do not have sufficient power to travel against the current [10, 11] and another work dealing with coverage path planning for a group of energy-constrained robots [20]. One notable work breaks from the tendency to emphasize complete coverage, instead attempting to conserve time and fuel by focusing coverage on regions of interest [13]. This allowed them to create a map of a coral reef area with half the distance coverage and power use than a lawnmower-style complete coverage algorithm would have required. Another paper, in which lawnmower-style coverage is applied to a Dubins vehicle, reformulates the problem as a variant of the Travelling Salesman Problem in order to obtain an optimal solution [12].

Even though there is substantial work done in the literature for area coverage, they do not address the coverage in tight and uneven spaces, such as rivers. Moreover, to the best of our knowledge, no one has addressed the efficiency problem of river navigation taking into account meanders information or any other river specific variables. Meanwhile, autonomous navigation on river can help reduce long and expensive exploration expeditions that scientists have to take for studying or monitoring rivers or marine environments in general.

In this work, we address the autonomous coverage problem for river surveying by taking advantage of the river meanders. We use the implicit information encoded in the shape of meanders regarding the speed of the river to choose a more efficient route when travelling downstream or upstream. In the following section, the problem is formally defined and the meander-based coverage approach is described with a suggestion for a possible improvement.

3 Proposed Method

The objective of this work is to perform complete coverage of a river segment by taking into account the varying speeds of water current across the river. We study the problem with an Autonomous Surface Vehicle (ASV) that is deployed on a known environment. The ASV is equipped with different types of depth sensors. We acquire the environment map from Google satellite images and convert it to an occupancy grid map $M : \mathbb{R}^2 \rightarrow \{0, 1\}$. Values of 0 indicate the portion of the river that is within the region of interest, while 1 indicates locations that we treat as obstacles. Assuming that the exact bounding box of the interest region is given together with starting point v_s, we can infer implicitly the general direction of the coverage.

3.1 Meander-Based Coverage (M-cover)

In the meander-based coverage, we are assuming that on the inner bend, the downriver speed of the current is slower from any neighbour region closer to the outside bend of the river. As it is shown in Fig. 2b along the passes that connect green dots the water flow is faster, whereas orange ones indicate region where the flow is slower. To find the meanders, we are looking into the intersection of two consecutive tangent lines to the curve of the river contour (Fig. 2a). If the lines intersect inside the river then inner curve is identified (orange vertex), otherwise if the intersection is on land then the outside bend is found (green vertex). Using this information the M-cover algorithm depicted in Algorithm 1 finds an efficient complete coverage path. It takes as input the map of the river M, a starting point v_s the spacing width information (sensor footprint size). First, the direction of the coverage is identified implicitly from v_s and M, then, the directional contour C_{vec} is generated (Line 2). Consequently, the river is split into S_{vec} segments, utilizing the meander information, using the above-explained intuition (Line 4). Each segment is split into segments that the robot can cover in a single pass (Line 6). We decompose area into even number of segments in order to return back to the initial starting point v_s. Each of the passes are assigned a direction: the first pass that is closest to the inner bank of the river is getting reserved for upwards travel, whereas the ones closer to the outer side are getting reserved for downwards travel (Lines 7–14). A pass is added between each consecutive segment of meanders: from orange to the closest green on the opposite edge of the river (Lines 16–17).

The simple M-cover approach does not take into account the change in the width of the river, which can affect the number of the passes one can generate. To solve that problem, we propose to adapt the L-cover algorithm introduced in previous work on river coverage [8]. With this modification, the algorithm will perform coverage in segments that have the same width (see Algorithm 2). In the same way, the width-based approach will take as an input the map M, the starting point v_s and the spacing information. In this case, we simply apply the L-Cover algorithm, to split the area into regions that have approximately the same width (Line 4), and then on each of those segments, we apply the M-Cover algorithm to generate the more efficient path. Fig. 3 shows the resulting clusters for the same segment of the river with different spacing value s.

Algorithm 1 M-Cover

Input: binary map of river M, starting point v_s, spacing parameter s
Output: a π path

```
$\Delta w \leftarrow$ initialize()
$C_{vec} \leftarrow$ getDirectionalContours($M$)
$\alpha \leftarrow$ getDownRiverDirection($C_{vec}, v_s$)
$S_{vec} \leftarrow$ getMeanderSegments($\Delta w, C_{vec}, \alpha$)
for each $S \in S_{vec}$ do
    $P \leftarrow splitIntoEvenPasses(S, C_{vec}, s)$
    $k \leftarrow |P|$
    for each $p_i, p_{k/2+i} \in P$ do
        if $p_i$ is on outside bend then
            $p_i \leftarrow$ down direction, $p_{k/2+i} \leftarrow$ up direction
        else
            $p_{k/2+i} \leftarrow$ down direction, $p_i \leftarrow$ up direction
            append $p_i, p_{k/2+i}$ to $\pi$
        end if
    end for
    $S_{prev} \leftarrow S$
    $p \leftarrow createPassBetween(S_{prev}, S)$
    append $p$ to $\pi$
end for
return $\pi$
```

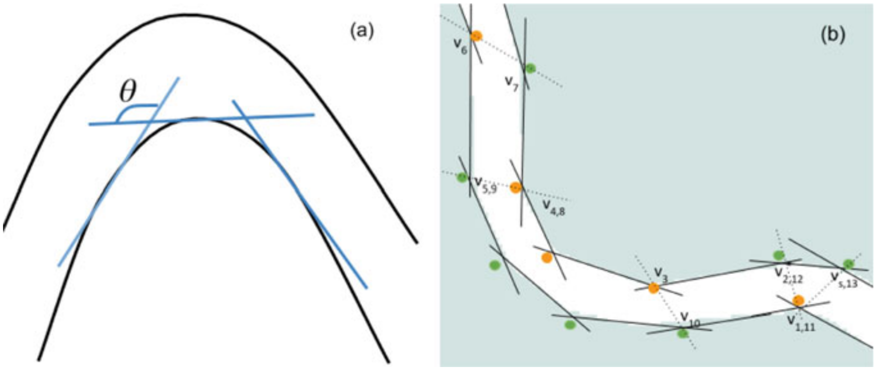


Fig. 2 A sketch for finding the meanders. **a** The procedure of checking the intersection of a neighbour pair of tangents. **b** The order of vertices the algorithm will visit if coverage is to be performed upwards

4 Results

The proposed M-cover algorithm was deployed with an ASV on different sizes of river segments, performing overall nearly 27km of coverage on the Congaree River, SC. We have used the AFRL jetyaks [14] equipped with a PixHawk controller for performing GPS-based waypoint navigation, a Raspberry Pi computer which runs

Algorithm 2 Width Based M-Cover

Input: binary map of river M, starting point v_s spacing parameter s
Output: a π path

```
$C_{vec} \leftarrow$ getDirectionalContours($M$)
$\theta \leftarrow$ getDownRiverDirection($C_{vec}, v_s$)
$Cl_{vec} \leftarrow getSameWidthClusters(C_{vec}, \theta, s)$
for each $Cl \in Cl_{vec}$ do
    $p \leftarrow$ M-Cover($Cl, v_s, s$)
    append $p$ to $\pi$
end for
return $\pi$
```

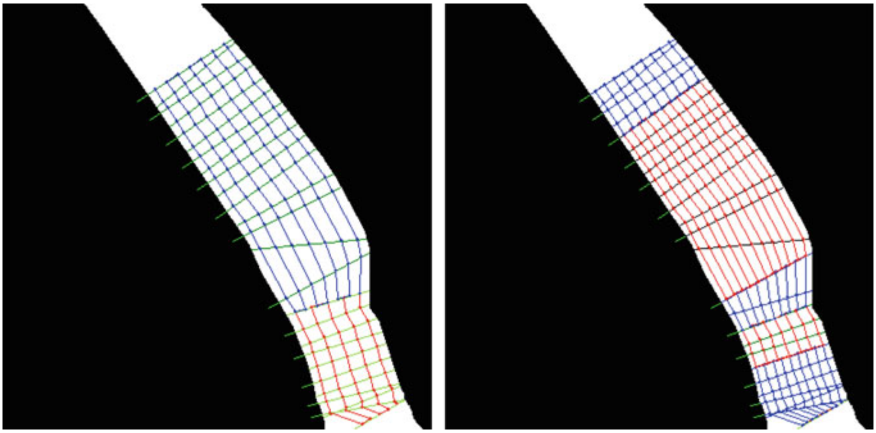


Fig. 3 An example of splitting an area into segments based on the width information

the Robot Operating System (ROS) framework [18] and records sensor data and GPS coordinates (Fig. 4).

The main objective of the experiments is to demonstrate that even with dynamically changing environments the proposed M-cover approach ensures more efficient coverage. We have deployed the ASV on 4.12 and 2.76 km segments of the Congaree river; see Table 1. The width of the river on average is 90m. The long segment was covered with small value of spacing which resulted in four passes (Fig. 7a), whereas in the smaller segment the ASV performed only two passes (Fig. 5). In addition, we run the L-cover algorithm with same spacing for the smaller region and similar to the M-cover two passes were generated (Fig. 6). For the longer region with the same spacing value, the L-cover generated segments with either three or five passes, though it resulted with a similar length of the coverage trajectory (Fig. 7b). When the execution time of the coverage operation of M-cover is compared with the L-cover for both experiments the M-cover is on average 20% more efficient. Note that results are only based on the performed field trials. It has been observed that the river current data changes even in an hourly base. Therefore, generating a graph model that will

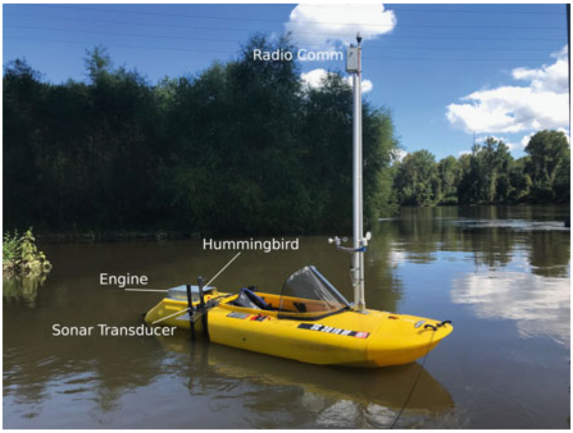


Fig. 4 The AFRL jetyak used during the field deployments. The main controller is the PixHawk that performs GPS-based waypoint navigation. In addition, different depth sensors are mounted on the boat for surveying operations

Table 1 The experimental results of deployments

Algorithm	Total area	Path length	Duration	# of passes
M-Cover	4.12km x 90m	16.6km	2h 55m	4
L-Cover	4.12km x 90m	16.3km	3h 35m	3 and 5
M-Cover	2.76km x 90m	5.32km	47m 36s	2
L-cover	2.76km x 90m	5.13km	59m 46s	2

represent the approximate currents would not be a comprehensive representation of a real-world scenario.

In addition, we have sampled small portions of the river and compared the coverage time for the two opposite banks of the river to show the effect of the current on coverage time. The results showed that when travelling upstream on the outside portion of the meander the coverage time is almost twice longer than if going downstream (approximately 47%).

In Fig. 7, we show the way-points generated by the different river coverage algorithms to illustrate the different patterns. If looking into complete coverage algorithms, we have shown in our previous work that L-cover is the more efficient approach, thus we compared the execution times of the proposed approach only with the L-cover approach. The T-cover approach fails mostly because of the cost of travelling back to the starting position, whereas the M-cover still performs coverage on the way back.

In addition from the data collected on the smaller river segment with a CruzPro DSP Active Depth, Temperature single ping SONAR Transducer, we have generated the depth map of the river segment using a Gaussian Process method (see Fig. 8).

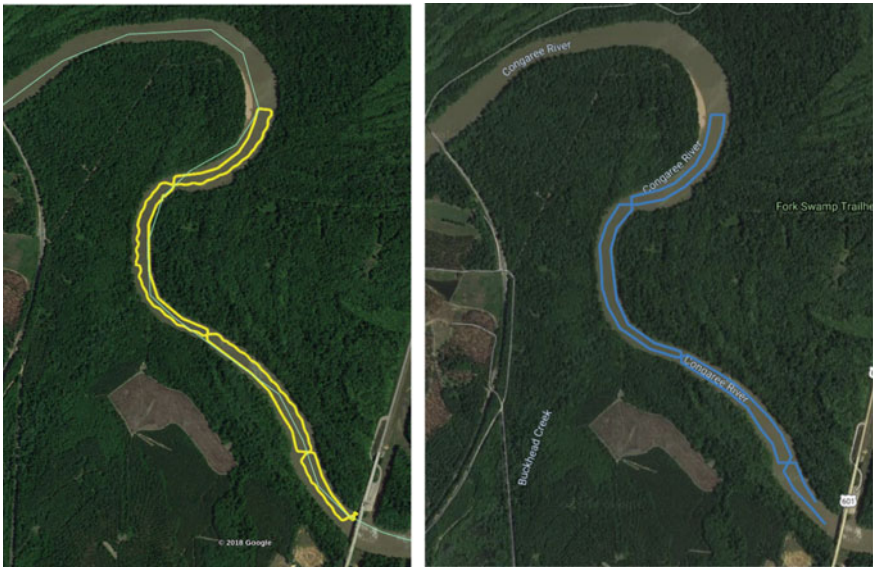

© 2018 Google
Congaree River
Congaree River
Fork Swamp Trailhe
Congaree River
Buckhead Creek
601

Fig. 5 The way-points generated by the M-cover algorithm (right) with the actual GPS trajectory (left) executed on the 2.76 km long river segment

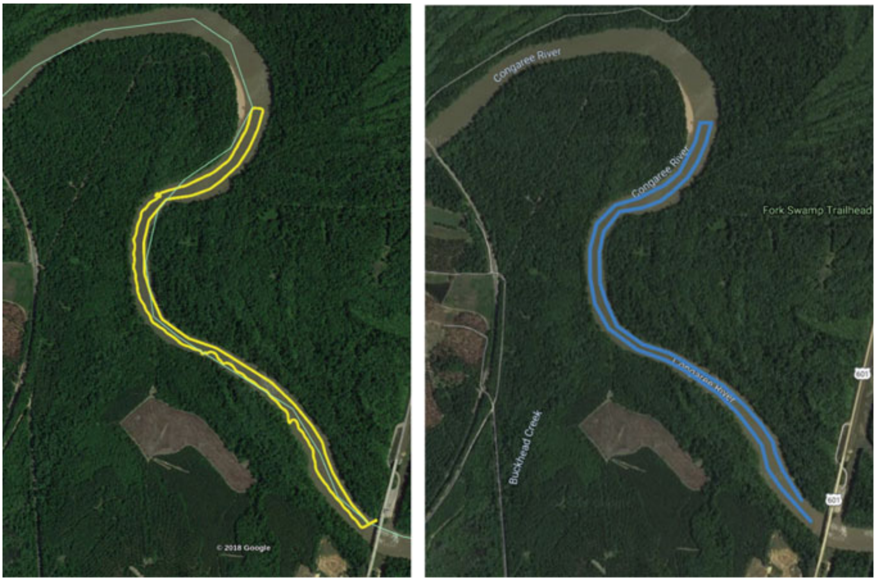

© 2018 Google
Congaree River
Congaree River
Fork Swamp Trailhead
Congaree River
Buckhead Creek
601
601

Fig. 6 The way-points for covering a river segment without taking into account meanders (right) with the actual GPS trajectory (left) executed on the 2.76 km long river segment

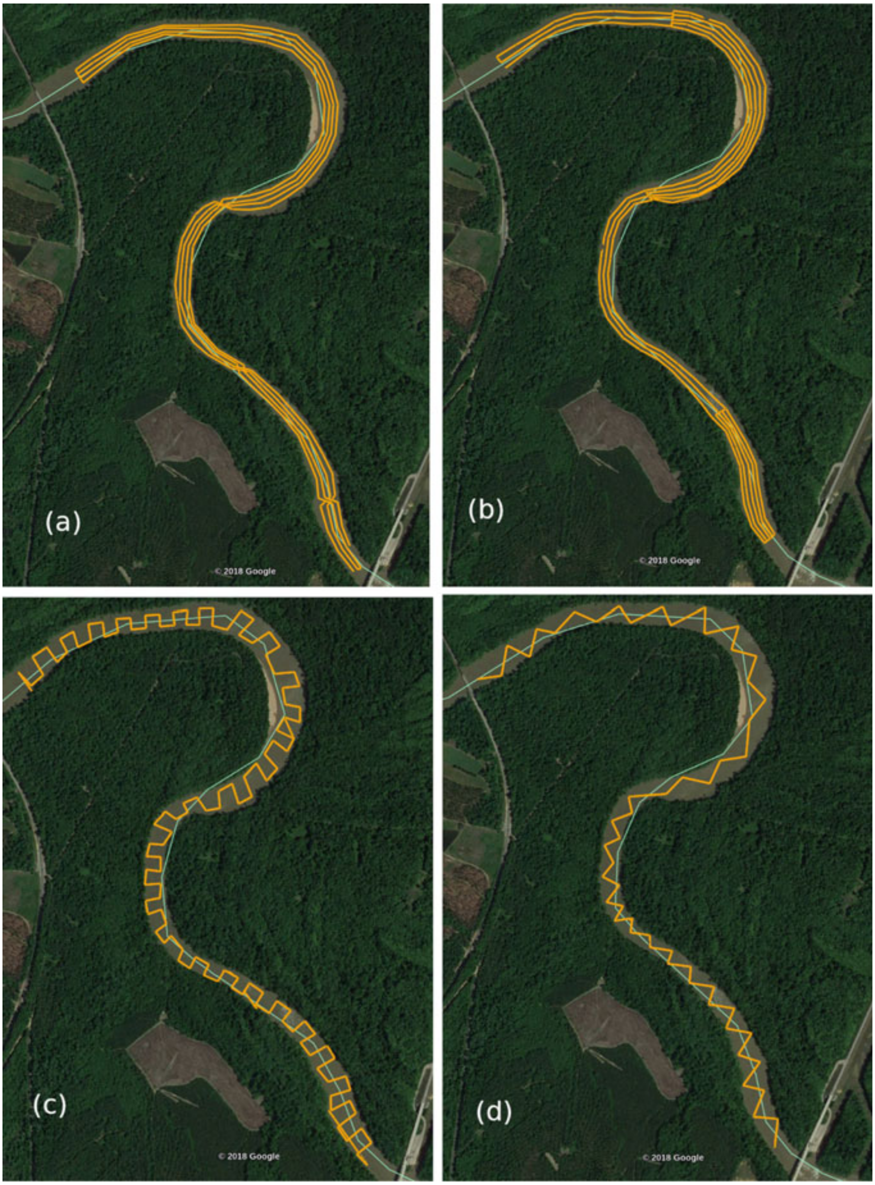

(a)
© 2018 Google
(b)
© 2018 Google
(c)
© 2018 Google
(d)
© 2018 Google

Fig. 7 Way-points of the different coverage patterns executed on the 4.2 km long segment of Congaree river, SC. **a** M-cover, **b** L-cover, **c** T-cover, **d** Z-cover

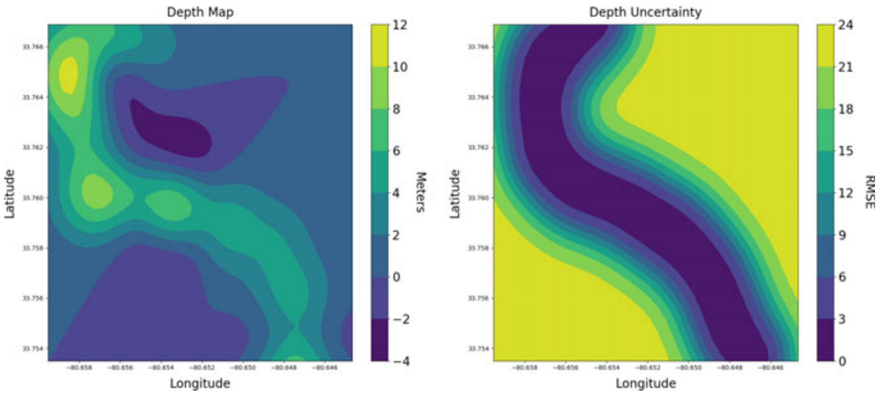


Fig. 8 Depth map generated by Gaussian process with RMSE on the left, indicating the accuracy of the prediction

5 Conclusion

The following work introduced a new coverage method specifically designed for rivers. It takes into account the connection of the bend of curve on meanders to choose the most optimal way of travelling along the river. The proposed M-cover method is a complete and more efficient coverage approach. We have demonstrated the improvements in the coverage performance by experimental results. The experiments were performed with an ASV on 4.12 and 2.76km long segments of the Congaree River, SC. In our previous work on river coverage, we showed that a complete coverage method termed L-cover is the most efficient. Therefore, we have deployed the ASV performing both the L-cover and the M-cover trajectories. The results demonstrated that the M-cover algorithm outperforms the L-cover approach by decreasing coverage time on average by 20%. In addition, we proposed a possible improvement of the M-cover by combining it with the L-cover algorithm to take into account also the width of the river, which can optimize also the number of passes generated per section. Finally, the depth map was generated using the data collected from CruzPro DSP single ping SONAR Transducer.

Taking into account the challenges encountered during the field deployments, obstacle avoidance strategies must be implemented for both underwater and above water obstacles. The important next contribution will be to extend this work to multiple robots [7, 9]. In addition given the dynamic changes of the environment, we are interested in generating a robust model of the water current of the river [15]. With this model path planning will associate different cost values and perform adaptive coverage.

Acknowledgements This work was partially supported by a SPARC Graduate Research Grant from the Office of the Vice President for Research at the University of South Carolina. The authors also would like to thank the National Science Foundation for its support (NSF 1513203).

References

1. Acar, E.U., Choset, H.: Sensor-based coverage of unknown environments: incremental construction of morse decompositions **21**(4), 345–366 (2002)
2. Acar, E.U., Choset, H., Rizzi, A.A., Atkar, P.N., Hull, D.: Morse decompositions for coverage tasks. Int. J. Robot. Res. **21**(4), 331–344 (2002)
3. Choset, H.: Coverage for robotics—a survey of recent results **31**(1–4), 113–126 (2001)
4. Einstein, A.: The cause of the formation of meanders in the courses of rivers and of the so-called baer's law. Die Naturwissenschaften **14**(11), 223–224 (1926)
5. Ferreira, H., Almeida, C., Martins, A., Almeida, J., Dias, N., Dias, A., Silva, E.: Autonomous bathymetry for risk assessment with ROAZ robotic surface vehicle. In: Oceans 2009-Europe, pp. 1–6. IEEE (2009)
6. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
7. Karapetyan, N., Benson, K., McKinney, C., Taslakian, P., Rekleitis, I.: Efficient multi-robot coverage of a known environment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1846–1852 (2017)
8. Karapetyan, N., Braude, A., Moulton, J., Burstein, J.A., Whitea, S., O'Kane, J.M., Rekleitis, I.: Riverine coverage with an autonomous surface vehicle over known environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2019)
9. Karapetyan, N., Moulton, J., Lewis, J.S., Quattrini Li, A., O'Kane, J.M., Rekleitis, I.M.: Multi-robot dubins coverage with autonomous surface vehicles. In: 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21–25, pp. 2373–2379 (2018)
10. Kwok, A., Martínez, S.: A coverage algorithm for drifters in a river environment. In: American Control Conference (ACC), pp. 6436–6441. IEEE (2010)
11. Kwok, A., Martínez, S.: Deployment of drifters in a piecewise-constant flow environment. In: 49th IEEE Conference on Decision and Control (CDC), pp. 6584–6589. IEEE (2010)
12. Lewis, J.S., Edwards, W., Benson, K., Rekleitis, I., O'Kane, J.M.: Semi-boustrophedon coverage with a Dubins vehicle. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5630–5637. IEEE (2017)
13. Manjanna, S., Kakodkar, N., Meghjani, M., Dudek, G.: Efficient terrain driven coral coverage using Gaussian Processes for mosaic synthesis. In: Computer and Robot Vision (CRV), 2016 13th Conference on, pp. 448–455. IEEE (2016)
14. Moulton, J., Karapetyan, N., Bukhsbaum, S., McKinney, C., Malebary, S., Sophocleous, G., Quattrini Li, A., Rekleitis, I.: An autonomous surface vehicle for long term operations. In: MTS/IEEE OCEANS, Charleston, pp. 1–6 (2018)
15. Moulton, J., Karapetyan, N., Quattrini Li, A., Rekleitis, I.: External force field modeling for autonomous surface vehicles. In: International Symposium on Experimental Robotics (ISER). Buenos Aires, Argentina (2018)
16. Njenga, K.J., Kwanza, J., Gathia, P.W.: Velocity distributions and meander formation of river channels. Int. J. Appl. **2**(9) (2012)
17. Qin, K., Shell, D.A.: Robots going round the bend—a comparative study of estimators for anticipating river meanders. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4934–4940. IEEE (2017)
18. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
19. Rekleitis, I., New, A.P., Rankin, E.S., Choset, H.: Efficient boustrophedon multi-robot coverage: an algorithmic approach. Annals Math. AI **52**(2–4), 109–142 (2008)

20. Sipahioglu, A., Kirlik, G., Parlaktuna, O., Yazici, A.: Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach. Robot. Auton. Syst. **58**(5), 529–538 (2010)
21. Snyder, F.D., Morris, D.D., Haley, P.H., Collins, R.T., Okerholm, A.M.: Autonomous river navigation. In: Mobile robots XVII, vol. 5609, pp. 221–233. International Society for Optics and Photonics (2004)

Autonomous 3D Semantic Mapping of Coral Reefs

Md Modasshir, Sharmin Rahman, and Ioannis Rekleitis

1 Introduction

Coral reefs play an integral part of the marine ecosystem and are home to numerous aquatic species [2, 28]. However, coral reefs are on a steep decline in health and population due to global warming and ocean pollution. Scientists predict that by 2100, global temperature will increase by 2–4.5 °C. Because of such alarming situation, marine biologists are vigilantly monitoring the coral reef with the help of scuba divers, and by deploying autonomous or human-operated vehicles, see Fig. 1 where a diver collects data over a transect. If divers examine the reefs, they usually follow a pre-specified transact and stop over a coral reef to record the health of corals as shown in Fig. 2. In the case of data collection by vehicles or automated methods, the experts first annotate the collected images for coral species and then analyze the recorded data to determine coral health. This monitoring process can be made more efficient by collecting visual data using autonomous or underwater vehicles and then only sending humans to analyze certain species if required. This efficiency can be achieved by utilizing a 3D semantic map of coral reefs which will provide useful information for both annotation and further study of the coral species. A 3D semantic map of a coral reef includes the 3D position of corals and their corresponding labels. Since the coral positions are known in such a map, scientists can navigate the same environment staying closer to certain species for detailed analysis. While

---

M. Modasshir (✉) · S. Rahman · I. Rekleitis
Computer Science and Engineering Department, University of South Carolina,
Columbia, USA
e-mail: modasshm@email.sc.edu

S. Rahman
e-mail: srahman@email.sc.edu

I. Rekleitis
e-mail: yiannisr@cse.sc.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_26

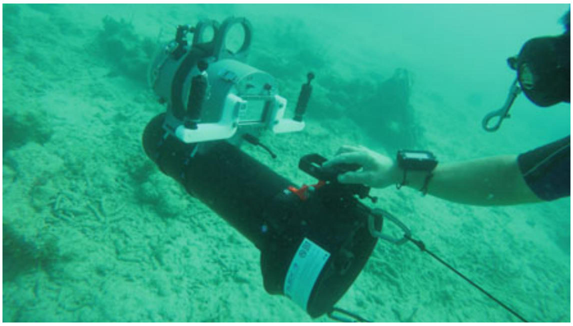


Fig. 1 Data collection over coral reefs, Barbados; Sensor Suite mounted on DPV

©Reef Recharge

Fig. 2 Expert observing and recording health of corals [1]

there are works to automate the annotation process [3, 19, 22], there are few efforts to incorporate coral species label information into the 3D maps.

Building a 3D semantic map of a coral reef will require two components: semantic information and a 3D map. The semantic information can be obtained using a coral detector. In recent times, deep learning has achieved tremendous success in detection [10, 17, 18, 27, 29]. A recent work by Modasshir et al. [23] proposed a CNN-based automated annotation and population counting system for coral reefs while moving in transacts. This approach used the RetinaNet [17] as a backbone classification network. In this approach [23], an image is passed through the RetinaNet for detection, thereby producing a bounding box annotation of corals. The detected corals are then tracked by a Kernelized Correlation Filter (KCF) [13] for a fixed number of successive frames, and then the detection is performed again to detect newer corals in the images. The authors utilized intersection-over-union (IOU) among newly identified and old tracked corals to determine whether corals were observed before to prevent a recount of those corals. Our work is inspired by [23], and RetinaNet is used for detection to create semantic labeling of corals in the images.

The next step in creating a 3D semantic map is building the 3D map by calculating 3D positions of observed features in subsequent images. In recent years, many vision-based real-time state estimation algorithms have been developed for indoor and outdoor environments providing robust solutions covering large-scale areas using monocular or stereo cameras, to name a few Simultaneous Localization and Mapping (SLAM) systems—ORB-SLAM [24], SVO [9], DSO [6], LSD-SLAM [7], and SVIn [26]. However, underwater environments suffer from low visibility, poor contrast, light and color attenuation, haze and scattering. Most of the state estimation algorithms often fail or show poor performances in such challenging scenarios. Joshi et al. [14] presented a comprehensive study and performance analysis of state-of-the-art open-source visual odometry algorithms. Among the vision-based SLAM systems, Direct Sparse Odometry (DSO) acquired the most accurate trajectory along with excellent 3D reconstruction in coral datasets collected by GoPro in Barbados. The DSO method [6] provides full photometric calibration, which accounts for lens attenuation, gamma correction, and known exposure times. Hence, in this paper, we augment the architecture of DSO by incorporating coral semantic labels according to the CNN-based detection network. The semantic information acquired from the CNN-based detection network is color-coded into the reconstruction by DSO. Once the 3D semantic map is built, the shape and size of an individual coral can be retrieved by a least-square shape fitting method suitable for different types of corals.

In this work, we propose an automated method for creating 3D semantic map only from visual information and estimating the coral population in the observed area. Our primary contributions are threefolds:

- Integrating CNN prediction into a point cloud generated by DSO to build a 3D semantic map.
- Calculating the volume of individual corals from features belonging to a coral.
- Estimating the coral population using detection and tracking.

We test our proposed method on two transacts in the Caribbean reef and show that our method reconstructs the 3D semantic map accurately and also counts the corals with accuracy on par with [23]. We also use an ellipsoid fitting method on a starlet coral to estimate its volume.

The next section discusses the works related to object detection and visual odometry. Section 3 illustrates the proposed methodology to combine CNN detection with DSO to build a semantic map. Experimental results from two transacts collected by GoPro cameras in Barbados are discussed in Sect. 4. We summarize the paper and discuss future research directions in Sect. 5.

2 Related Work

2.1 Object Detection

There are several works in coral classification in the literature. Most of the traditional approaches focused on using pixel-based information and textural appearances [20, 21, 25, 31]. Early work by Beijbom et al. [4] proposed to use color descriptors and texture at multiple scales. The authors proposed a Maximum Response filter bank for color and texture feature extraction. Each color channel was passed through the filter separately, and the filter responses were stacked. The stacked response was further passed through a Support Vector Machine with Radial Basis Function kernel to train the model. Other than the traditional approaches, there are a few works using deep learning for coral classification. Mahmood et al. [19] proposed a method combining learned features and hand-crafted features from multi-scale patches. The authors extracted the learned features from the last convolutional layer of VGGNet [30]. These learned features and the handcrafted features were passed through a multi-layer perceptron classifier. In previous work [22], we proposed a densely connected CNN for coral classification. A point annotated dataset [4] was used and patches were extracted, centered on point annotation. These patches are fed to the classifier where the patches were cropped three times at different sizes keeping the same center. State-of-the-art performance was achieved for coral classification. This paper extends our work in Modasshir et al. [23] as the dataset used contains severe class imbalance.

2.2 Visual Odometry

Vision-based SLAM systems can be categorized into direct and indirect (feature-based) method. Indirect methods (e.g., ORB-SLAM [24],OKVIS [15], SVIn [26]) include a pre-processing step accounting for the detection and tracking of features (e.g., corners) in consecutive images and optimizing the *geometric error*. Direct methods (e.g., DSO [6], LSD-SLAM [7]), on the other hand, consider pixel intensity and optimize the *photometric error* based on the direct image alignment without the need of any pre-processing step. Intermediate method, i.e., *semi-direct* also exists (e.g., SVO [9]) which combines both direct and indirect methods. While direct methods provide a dense or semi-dense representation of the environment, they often fail due to the *brightness consistency* assumption in low contrast environment with numerous lighting variations. As feature-based methods use photometric invariant features, they show a better estimation of the pose but as a result of using only a sparse set of *keypoints* the 3D reconstruction of the surroundings is very sparse. Direct methods suffer heavily from large rotational change and slow initialization; however, the reconstruction by direct methods is less sparse compared to that of feature-based methods. DSO [6] has shown promising performance both in terms of tracking and 3D reconstruction. Instead of considering a dense formulation, DSO selects a sparse set of gradient-rich pixels in the image which retains excellent reconstruction along with accurate pose estimation. In this work, we have utilized DSO as it has provided the best reconstruction with the most accurate trajectories. As the field of Visual SLAM evolves, the proposed methodology can be extended to the latest products.

3 Approach

The proposed method works with videos or sequences of images. Therefore, the corals detected can be tracked over subsequent images which reduce the frequency of running the detection algorithm. This reduction makes the system faster and thus capable of running online. There are three steps of the proposed method: detection, tracking, and semantic mapping. For a sequence of images, f_1 to f_{n-1}, we acquire the location of corals in the image, f_0 from the detection system. These locations are provided as bounding box, termed Regions of Interest (ROI), and later are utilized to initialize the KCF tracker in order to track the observed corals in the next f_1 to f_{n-1} frames before the detection runs again on f_n. The newly detected corals in f_n are then matched against already tracked locations for overlap. If there is significant overlap among new ROIs and tracked ROIs, then merely the locations of tracked ROIs are updated to the newly detected ROIs. Otherwise, the system initializes new instances of KCF tracker with the new ROIs. In parallel to tracking, the ROIs for each frame are also passed to DSO which uses the ROIs to save the semantic information of the 3D points by color-coding them while creating the 3D map (Fig. 3).

3.1 Detection

RetinaNet: RetinaNet [17] is a one-stage detector that includes a backbone network and two subnetworks. The backbone network is used to extract generic learned features from images. The Feature Pyramid Network (FPN) [16] is chosen as the backbone network. The FPN takes a single resolution image as input and then creates multi-scale features to augment a standard CNN with a top-down pathway and lateral connections. This use of multi-scale features enables the network to detect objects of different sizes. The FPN network can use various networks as a base network such as VGGNet [30], ResNets [12]. In our system, we choose ResNet with 50 layer variation as the base network of the FPN. The two subnetworks of the RetinaNet are utilized to classify and regress the bounding box locations. The final layer of the RetinaNet network is redesigned to accommodate eight classes of corals.

Focal Loss: The classification subnet of the RetinaNet is optimized using focal loss [17]. The focal loss is specifically suited for our classification problem as the focal loss is designed to handle a high-class imbalance in the training dataset. For a target class, k with an estimated probability $s_k \in [0, 1]$, we define the focal loss as

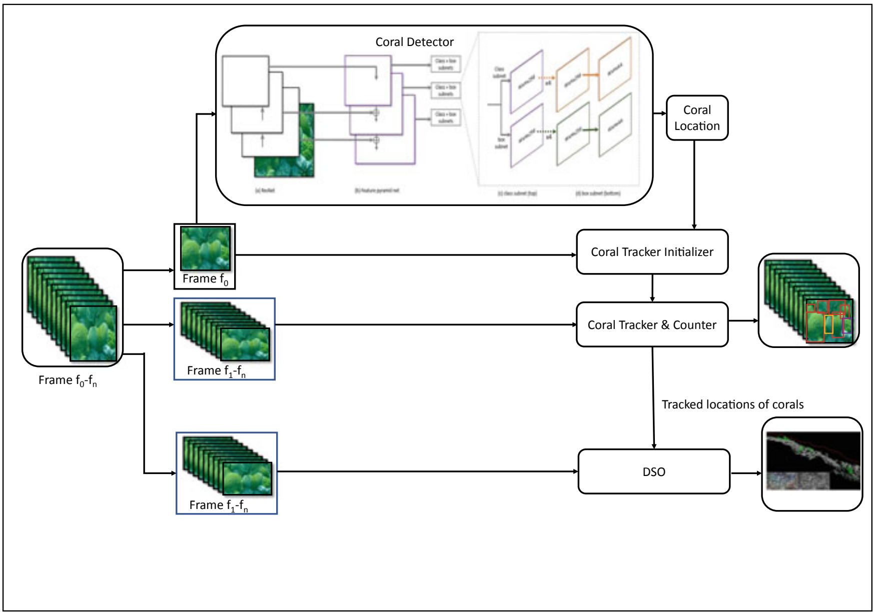

Coral Detector
Coral Location
Frame f0
Coral Tracker Initializer
Frame f0-fn
Frame f1-fn
Coral Tracker & Counter
Tracked locations of corals
Frame f1-fn
DSO

Fig. 3 Overview of the proposed approach

$$loss(s_k) = -\alpha\,(1 - s_k)^{\gamma}\,log(s_k)$$

where $\gamma \geq 0$ is a focusing parameter which can be tuned and $\alpha \in [0, 1]$ is a weighting factor. Because of these parameters, γ and α, a lower loss is assigned to easily classifiable examples, thereby an overall update of hyper-parameters in the network can focus on hard instances in the dataset. We choose the inverse of samples per class as the weighting factor of α. The γ controls how smoothly the loss for easily classifiable examples is down-scaled.

Smooth L1 Loss: The bounding box regression or ROIs subnet is optimized using smooth L1 loss. For predicted ROIs p and ground truth ROIs g, the smooth L1 loss is

$$Loss_{roi} = \sum_{i \in x,y,w,h} smooth_{L_1}(p_i - g_i)$$

where x and y are the top-left coordinates of the ROI, and w and h are width and height of the ROI. The function $smooth_{L_1}$ is defined piece-wise as

$$smooth_{L_1}(q) = \begin{cases} 0.5\,q^2, \text{ if } |q| < 1. \\ |q| - 0.5, \text{ otherwise.} \end{cases} \tag{1}$$

Training: The pretrained weights on ImageNet dataset [5] were used to initialize the base network, ResNet of the FPN network. We initialize all other layers to zero-mean

Gaussian distribution with a standard deviation of 0.01. The hyper-parameters were optimized by stochastic gradient descent (SGD). The training epochs are 150 with 0.001 initial learning rate and $1e^{-5}$ decay.

3.2 Tracking

The KCF [13] tracker utilizes multiple channels of color images to improve a correlation filter. Let the Gaussian shaped response be $r = [r_1, r_2, \ldots, r_j]^T \in \mathbb{R}$ and the input vector be $c_d \in \mathbb{R}^{j\times 1}$.

The filter weights w of the KCF are updated by optimizing:

$$\dot{w} = arg \min_{w} \sum_{j=1}^{J} (r_j - \sum_{d=1}^{D} R_{j,d}^T W_d)_2^2 + \lambda \, \|W\|_2^2 \tag{2}$$

where $R_{j,d}$ is the j-step circular shift of the input vector R_d, r_j is the j-th element of r, $W = [W_1^T, W_2^T, \ldots, W_D^T]^T$ where $W_d \in \mathbb{R}^{J*1}$ refers to the filter of the d-th channel [32].

3.3 Counting

For a number of frames, f_1 to f_{n-1}, the CNN-based detector localizes corals by providing bounding boxes in the frame f_1. The KCF tracker instances are initialized to these bounding boxes and then tracked in the frames from f_2 to f_{n-1}. The detector performs bounding box prediction again on the f_n frame. The predicted bounding boxes are then matched against KCF tracked bounding boxes using Intersection-over-Union (IOU) [8] to prevent a recount. To be considered as a new coral object, the bounding box prediction, R_p and the tracked bounding box R_t must have overlap ratio M_o less than 0.5. The overlap ratio, M_o, is defined as

$$M_o = \frac{area(R_p \cap R_t)}{area(R_p \cup R_t)}, \tag{3}$$

where $R_p \cap R_t$ indicates the intersection between predicted and tracked bounding boxes and $R_p \cup R_t$ indicates their union. If the overlap ratio, M_o, is below 0.5, the detected coral counts as a new coral object and a new instance of KCF tracks the new coral. Otherwise, KCF tracked bounding box coordinates are updated to that of the CNN detector produced bounding box.

3.4 Semantic Mapping

DSO: DSO provides a *direct* and *sparse* formulation for a monocular visual odometry system by combining the benefits of the direct approach and the flexibility of sparse approaches, i.e., efficient, joint optimization of all model parameters including the inverse depth in a reference frame, camera motion, and camera intrinsics. By sampling from pixels across all image regions with a high-intensity gradient (e.g., edges) and omitting the smoothness prior used in other direct methods, DSO is capable of real-time tracking. DSO accounts for the full photometric calibration which leads to accurate and robust state estimation.

Like the more recent Visual Odometry (VO) or Visual Inertial Odometry (VIO) systems, DSO follows windowed optimization and marginalization. A window of a fixed number of active keyframes is maintained. DSO provides a fully direct probabilistic model by continuous optimization of the photometric error over a local window of the current frames. The geometry of a 3D point is presented only by one parameter—the inverse depth in the reference frame. The photometric error of a point is represented as the weighted SSD over a small neighborhood of pixels, thus providing a good trade-off between computational requirement and robustness to the motion blur. Keyframe tracking is based on the two-frame direct image alignment. New keyframes are created if the field-of-view changes or the translation causes occlusions and disocclusions. Keyframe marginalization takes place when the active set of variables becomes too large; at first, all the points belonging to that keyframe as well as points that have not been observed in the last two keyframes are marginalized. Initialization in DSO is critical and slow. Being monocular, DSO requires low translational change and cannot handle sizeable rotational change.

Integrating CNN and DSO: Once DSO initializes and starts building the map, the detection and tracking algorithms begin their operations on a parallel thread. For each frame, DSO is only modified to utilize semantic information provided as bounding boxes by the detection or the tracking pipeline to color-code the reprojected feature-points in the point cloud.

3.5 Coral Volume Estimation

There are different shape fitting methods. We consider the ellipsoid fitting method as ellipsoids are particularly suitable for the most common corals in the collected data: starlet and brain. We follow the algebraic ellipsoid fitting method. Generally, nine parameters are required to define an ellipsoid. The required parameters are three coordinates of the center, three semi-axes a, b, c and three rotational angles. Given data points, P, unknown ellipsoid parameters, X and a design matrix A, we have

$$A_{n,u}.\delta X_u = l_n \tag{4}$$

where n is the number of data points, u is the number of unknown parameters. For this equation to have a solution, we need $u \leq n$. We choose the l_2-norm method to solve the system of linear equations. Once we have the semi-axes a, b, c, the volume V is

$$V = \frac{4}{3}\pi abc \tag{5}$$

4 Experimental Results and Discussion

To validate our approach, we have selected two different trajectories: 3 min 27 s (henceforth, referred to as the 3 min trajectory) trajectory with a length of 40.29 m and a 10 min trajectory with a length of 315.39 m. Both trajectories were collected by a GoPro camera over live coral reefs. The 3 min trajectory data were collected by a scuba diver while the 10 min trajectory data were collected by utilizing an underwater scooter. The CNN detector was trained to detect the following corals of seven types: Brain, Mustard, Star, Starlet, Maze, Sponge, Finger, and Fire coral. However, in both trajectories, Finger and Fire coral are absent. Therefore, we do not report count for these two types of corals.

The color codes used to represent different corals in the 3D semantic map is shown in Table 1. Figure 4 shows the 3D semantic map of the 3 min trajectory. The point cloud clearly shows the coral features belonging to an individual coral are close together in 3D space and forms the shape of the coral. Figure 5 shows two pairs of observed image and their corresponding zoomed-in semantic map. As can be seen from Fig. 5, the reconstructed 3D point cloud retains the shape and size of the observed corals well. It is worth noting that the 3D point cloud is constructed using many successive frames, and therefore, does not correspond precisely to the location of the bounding box predictions in the raw images in Fig. 5.

Figure 6 shows the 3D semantic mapping of the 10 min trajectory. Similar to the 3 min trajectory mapping, the coral features are located nearby in 3D space and preserve the shape of the corals in the 3D mapping. However, the reconstruction could have been improved if loop closure was used along with the DSO method. Due to the absence of loop closure, we found the coral features being duplicated and positioned in different 3D space when the camera revisits the same place. It is worth noting that we tried Direct Sparse Odometry with Loop Closure [11] and the loop closure did not work. Most of the SLAM systems fail to produce accurate trajectories on both of our transacts. In the future, we plan to use SVIn [26], which

Table 1 Color codes used in 3D semantic mapping for different types of corals

	Brain	Mustard	Star	Starlet	Maze	Sponge
Color	Green	Purple	Teal	Magenta	Aqua	Blue

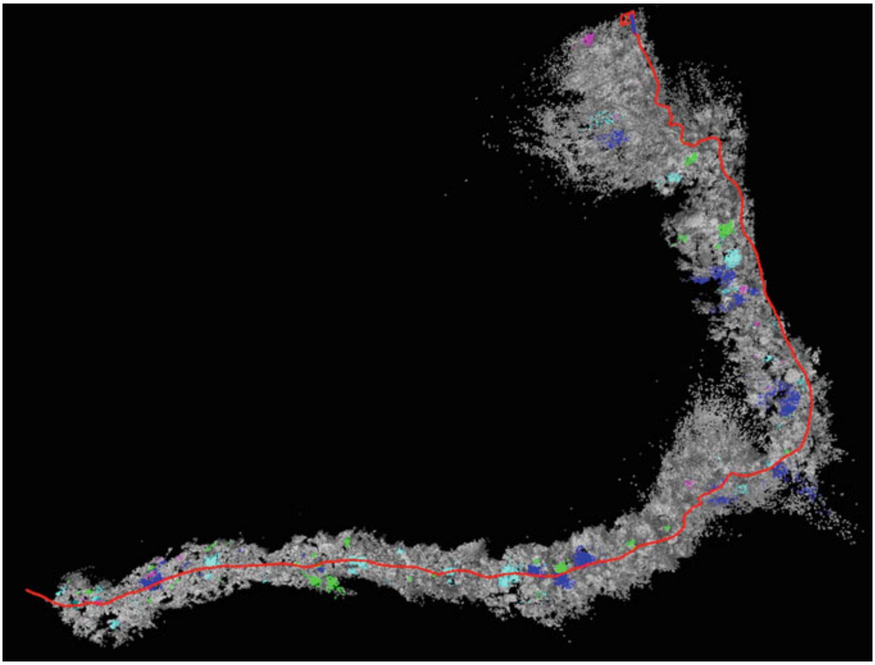


Fig. 4 3D semantic map of the 3 min trajectory. Features from different corals are displayed in different colors according to Table 1

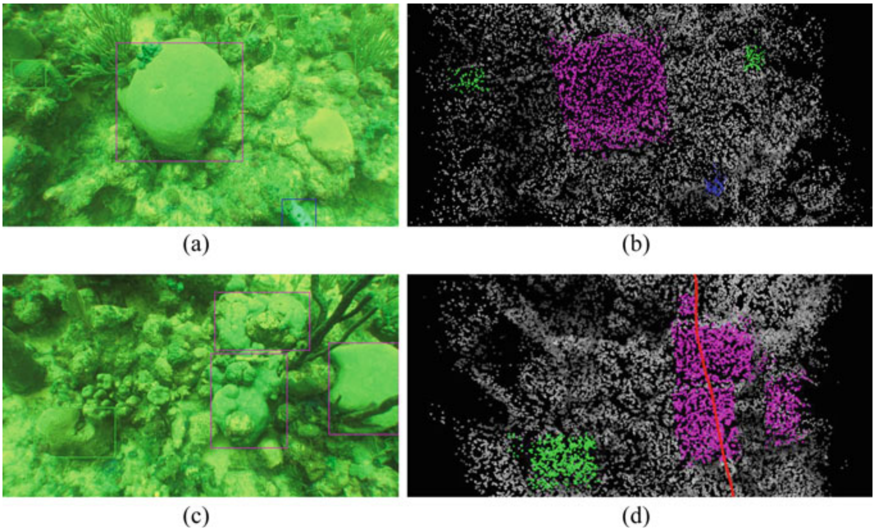


Fig. 5 Detail of the 3D semantic map of Fig. 4. **a**, **c** are the raw images **b**, **d** are the detected features in 3D color coded. In **a** the main object is a starlet coral (marked in magenta), and in **c** there are some sponges and a few brain corals (marked in blue and green accordingly)

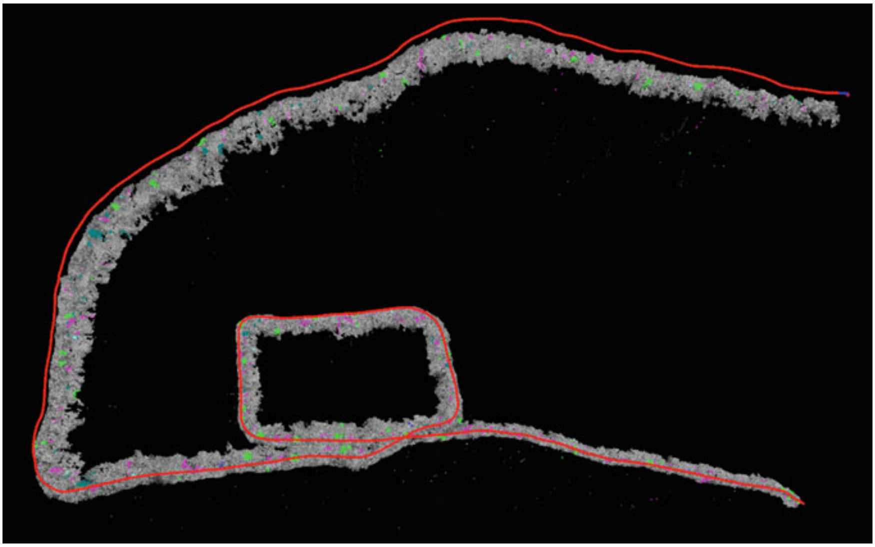


Fig. 6 3D semantic map of the 10 min trajectory. Features from different corals are displayed in different colors according to Table 1

fuses sonar and vision sensors, to acquire precise ground truth trajectory for better reconstruction.

Figure 7 shows two pairs of close-up semantic reconstruction along with corresponding raw images and the predicted bounding boxes. Figure 7c shows a critical case for the rebuilding of the semantic map. When the bounding boxes of different corals overlap, the color-coding process cannot retain spatial information appropriately, therefore, also loses the shape and size data of the coral. This challenge comes inherently with the use of bounding box prediction and can only be overcome by CNN-based semantic segmentation.

4.1 Counting

Quantitative results are reported in Table 2 for both trajectories for four types of corals: Brain, Mustard, Star, Starlet, Maze, and one non-coral type: Sponge. We report the count of each kind of coral objects by our prediction and tracking system as well as human-annotated count. The counting system performs favorably well when compared to the work of Modasshir et al. [23]. Empirically, the detection performed relatively well on the 10 min trajectory compared to the 3 min trajectory, because the 3 min trajectory images have greenish appearance due to severe red-channel suppression which is a known phenomenon in underwater photos.

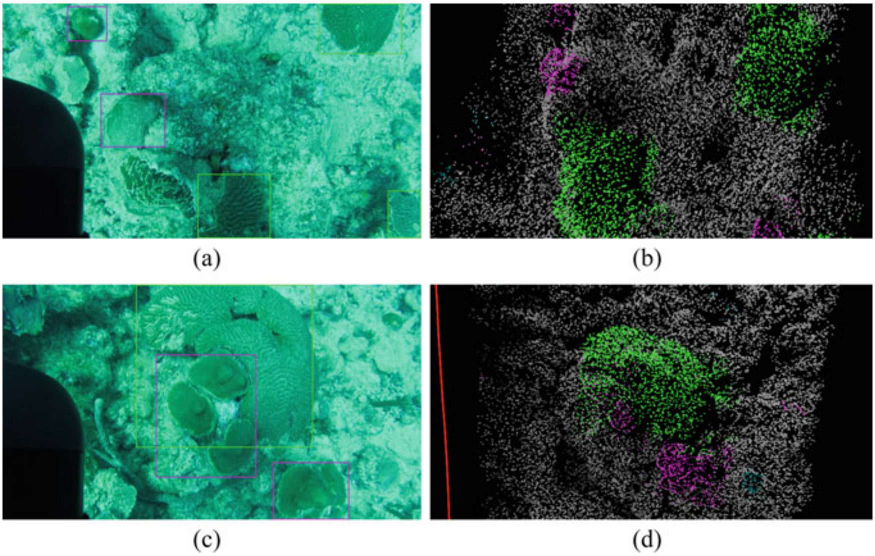


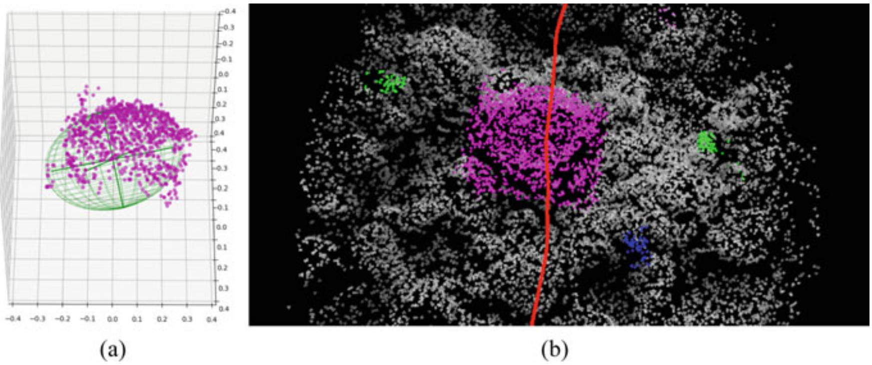
Fig. 7 Detail of the 3D semantic map of Fig. 6. **a**, **c** are the raw images **b**, **d** are the detected features in 3D color coded. In **a** two starlet coral and three brain corals are detected (marked in magenta and green respectively), and in **c** there are one brain coral and two starlet coral detected (marked in green and magenta in order)

Table 2 Coral counting for different trajectories. CNN-prediction/Human-Annotated

	Brain	Mustard	Star	Starlet	Maze	Sponge
T1:3 min	31/37	0/2	5/7	37/43	11/15	17/17
T2:10 min	90/97	39/47	68/75	161/176	26/28	5/6

4.2 Volumetric Coral Evaluation

Different types of corals require different shape fitting methods. Starlet and Brain corals are the most common types in our trajectories. Both coral types are of ellipsoid shape. We show ellipsoid fitting on a starlet coral in Fig. 8. The features corresponding only to the starlet coral in the point cloud (shown in magenta) are used for the ellipsoid fitting. The calculated radii of the ellipsoid are 0.24, 0.26, 0.31 m which closely matches our empirical observation of the starlet coral. The volume of the starlet coral using Eq. 5 is 0.081 m^3. In future work, we plan to integrate different shape fitting methods suitable for different types of corals.



(a)

(b)

Fig. 8 Ellipsoid fit on the features from starlet coral. **a** shows the result of ellipse fitting and **b** displays corresponding features (in magenta) in the point cloud

5 Conclusion

In this paper, we presented a system for building a 3D semantic map of a coral reef in a transect using only visual information as well as estimating the coral population in the transact. In our proposed method, we showed how to incorporate semantic information retrieved from CNN-based detector into a SLAM system to create an enriched 3D semantic map. The proposed approach will enable marine scientists to monitor and assess the health of coral reefs of a much larger area swiftly. By retrieving the shape of corals from transects over time in specific areas, our method will also allow evaluating erosion of corals.

While visual information helped create a 3D semantic map, more sensor fusion will make the 3D semantic map more accurate by helping in the localization of 3D points. Future work will investigate how to integrate 3D points belonging to a coral object into tracking the coral, thereby, replacing the KCF tracker and making the system even faster.

Acknowledgements This work was made possible through the generous support of National Science Foundation grants (NSF 1513203).

References

1. Coral reef monitoring|reef recharge. http://reefrecharge.com/coral-reef-monitoring/. Accessed 18 April 2019
2. Corals of the world—variation in species. http://coral.aims.gov.au/info/taxonomy-variation.jsp. Accessed 30 November 2017
3. Beijbom, O., Edmunds, P.J., Kline, D., Mitchell, B.G., Kriegman, D., et al.: Automated annotation of coral reef survey images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1170–1177 (2012)

4. Beijbom, O., Edmunds, P.J., Kline, D.I., Mitchell, B.G., Kriegman, D.: Automated annotation of coral reef survey images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1170–1177 (2012)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR09 (2009)
6. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. **40**(3), 611–625 (2018)
7. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM, vol. 8690, pp. 834–849. Springer Int. Publishing (2014)
8. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (2015)
9. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D.: SVO: semidirect visual odometry for monocular and multicamera systems. **33**(2) (2017)
10. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: DSSD: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
11. Gao, X., Wang, R., Demmel, N., Cremers, D.: LDSO: Direct sparse odometry with loop closure. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2198–2204. IEEE (2018)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
14. Joshi, B., Rahman, S., Cain, B., Johnson, J., Kalitazkis, M., Xanthidis, M., Karapetyan, N., Hernandez, A., Quattrini Li, A., Vitzilaios, N., Rekleitis, I.: Experimental comparison of open source vision-inertial-based state estimation algorithms (2019) (under review)
15. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. **34**(3), 314–334 (2015)
16. Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR, vol. 1, p. 4 (2017)
17. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37. Springer
19. Mahmood, A., et al.: Coral classification with hybrid feature representations. In: IEEE International Conference on Image Processing (ICIP), pp. 519–523 (2016)
20. Marcos, M.S.A., David, L., Peñaflor, E., Ticzon, V., Soriano, M.: Automated benthic counting of living and non-living components in ngedarrak reef, palau via subsurface underwater video. Environ. Monitor. Assess. **145**(1–3), 177–184 (2008)
21. Mehta, A., Ribeiro, E., Gilner, J., van Woesik, R.: Coral reef texture classification using support vector machines. In: International Conference on Computer Vision Theory and Applications (VISAPP), pp. 302–310 (2007)
22. Modasshir, M., Li, A.Q., Rekleitis, I.: Mdnet: Multi-patch dense network for coral classification. In: MTS/IEEE Oceans Charleston, Charleston, SC, USA (Oct. 2018) (accepted)
23. Modasshir, M., Rahman, S., Youngquist, O., Rekleitis, I.: Coral identification and counting with an autonomous underwater vehicle. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), Kuala Lumpur, Malaysia, pp. 524–529 (2018)
24. Mur-Artal, R., Montiel, J., Tardos, J.: ORB-SLAM: a versatile and accurate monocular SLAM system, vol. 31, pp. 1147–1163 (2015)
25. Pizarro, O., Rigby, P., Johnson-Roberson, M., Williams, S.B., Colquhoun, J.: Towards image-based marine habitat classification. In: OCEANS 2008, pp. 1–7. IEEE (2008)
26. Rahman, S., Li, A.Q., Rekleitis, I.: Sonar visual inertial SLAM of underwater structures. In: IEEE International Conference on Robotics and Automation, Brisbane, Australia, pp. 5190–5196 (May 2018)

27. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
28. Rogers, C., Garrison, G., Grober, R., Hillis, Z., F.ie, M.: Coral reef monitoring manual for the Caribbean and western Atlantic, 110 pp. Virgin Islands National Park Ilus (1994)
29. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
31. Stokes, M.D., Deane, G.B.: Automated processing of coral reef benthic images. Limnol. Oceanogr. Methods **7**(2), 157–168 (2009)
32. Sun, C., Wang, D., Lu, H., Yang, M.H.: Correlation tracking via joint discrimination and reliability learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 489–497 (2018)

Dynamic Autonomous Surface Vehicle Controls Under Changing Environmental Forces

Jason Moulton, Nare Karapetyan, Michail Kalaitzakis, Alberto Quattrini Li, Nikolaos Vitzilaios, and Ioannis Rekleitis

1 Introduction

As the demand for data collection and monitoring continues to expand across all reaches of the globe, research and development of Autonomous Surface Vehicles (ASVs) control in uncertain environments is essential. While the tasks and missions to which an ASV could be assigned are only limited by one's imagination, our desire to explore the unexplored increases the capabilities required in an ASV. One such hypothetical employment for an ASV would have been to assist with monitoring and recovery after the Fukushima Daiichi nuclear disaster following the 2011 earthquake in Japan (Fig. 1).

---

[1] https://www.flickr.com/photos/iaea_imagebank/10722882954.

J. Moulton (✉) · N. Karapetyan · I. Rekleitis
Computer Science and Engineering Department, University of South Carolina,
Columbia, USA
e-mail: moulton@email.sc.edu

N. Karapetyan
e-mail: nare@email.sc.edu

I. Rekleitis
e-mail: yiannisr@cse.sc.edu

M. Kalaitzakis · N. Vitzilaios
Department of Mechanical Engineering, University of South Carolina,
Columbia, USA
e-mail: michailk@email.sc.edu

N. Vitzilaios
e-mail: vitzilaios@sc.edu

A. Quattrini Li
Computer Science Department, Dartmouth College, Hanover, Germany
e-mail: alberto.quattrini.li@dartmouth.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_27

Fig. 1 Fukushima Daiichi nuclear complex in Okumamachi, Japan taken during water sampling and analysis by IEAE in 2013

For a less catastrophic scenario, with over 3.5 million miles of rivers in the United States alone, the ability to access, cover, and navigate them requires an ASV with long-range potential, as well as a precise trajectory following capability to ensure safe maneuvers. In addition, the ability to take into account the effect of external forces would improve the efficiency in planning for coverage as well as savings in power and fuel consumption. While there exists much research into the effects of natural phenomena such as wind and current in ocean areas, there remains a void when it comes to studying the same type of effects on smaller ASVs in confined areas with higher currents such as rivers. Operating in the air and water domains simultaneously exposes ASVs to wind and current external forces that can easily overwhelm current Proportional, Integral, Derivative (PID) controlled navigation systems.

Our paper pushes the research boundaries to advance the state of the art which will allow ASVs to be utilized in increasingly challenging conditions to ensure that ASVs become ubiquitous with researchers, engineers, and environmental scientists.

1.1 Problem Definition

Addressing the challenge of operating in the presence of non-trivial external forces can be done in two different scales. If a long-range map of the external forces is available then large scale planning can take the effects of the external forces into account. For example, coverage planning algorithms [5, 6] can include the force map as an input variable in order to improve mission planning. In a smaller scale, real-time force measurements can be used in a reactive controller to accurately track the desired trajectory. In analogy, knowing the traffic patterns in a city can generate routes through less congested streets, while a driver sensing slipping on ice, or pushed by a wind gust can guide the vehicle accordingly. In this paper, we provide a novel method for augmenting a controller with information from local disturbances. A manifestation of this problem is illustrated in Fig. 2, where the ASV is unable to maintain an accurate trajectory due to the PID controller being overcome by the changing currents.

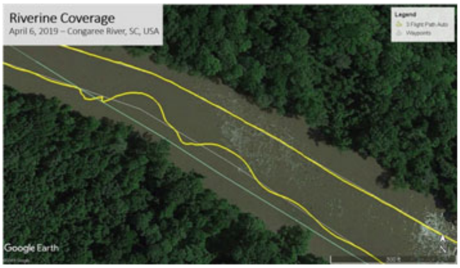


Fig. 2 Target trajectory unable to be followed in the downstream pass due to high 3.0 m/s currents in the area

In most riverine environments, it should be noted that a standard PID-driven way-point navigation controller can be tuned to maintain course either when moving with or against the external force, but not both conditions with the same gains. See, for example, Fig. 2, where the trajectory is followed accurately upstream, meaning against the current, and the erratic trajectory is produced from a downstream path.

1.2 Related Work

There have been several approaches in developing small autonomous surface vehicles. Different combinations of single hull, twin-hull, electric, or gas combustion can be observed in several publications [1–3, 9]. Rodriquez et al. present a comprehensive feasibility review of ASVs as of 2012 [16]. Recently, Woods Hole Oceanographic Institute (WHOI) [7] created an ASV based on Mokai's gas-powered kayak. The capability for longer duration and increased payload led us to utilize the same base platform for our ASV [11].

Rasal of Santa Clara University presented early work in 2013 on the building of their SWATH ASV and applied some strategies for overcoming environmental disturbances based on their effects measured by an integrated IMU [15]. While their results were improved over the standard way-point navigator, control based on reactive measurements does not accomplish our goal of path following in faster moving currents and winds. However, their off-board control system inspires our design and implementation for customized control sequences for future mission-specific tasks in longer range, more volatile environments.

Using the platform and the integrated sensor suite presented in prior work [6, 11], we developed methods for measuring wind and current acting on the ASV. This led to the ability to predict wind and current forces using a Gaussian Process for short temporal periods. In addition, this collection of measurements can enable potentially both optimal mission planning as well as discrete feed-forward control development [4].

Controller development research mainly relies on models using inertial tracking and compensating for roll, pitch, and yaw rates to provide course corrections. Our approach is more proactive, in that we actively measure and model environmental variables and provide the course correction prior to being swept off course.

Related research focused solely on minimizing error tracking includes work from Tsu-Chin [17] and robust digital tracking control based on a disturbance observer from Lee et al. [8]. These works closely model our setup, except that, in our case, the wind and current sensors are taking on the role of the disturbance observers. Pereira et al. focused on position anchoring of small under-actuated ASVs in windy conditions [13]. Their performance was good in conditions the authors admitted to being moderate. So, in order to enable accurate tracking in volatile conditions, we will extend our tracking control research to be proactive. The two essential steps in completing this work are first, modeling the effects of winds and currents on an ASV, followed by using this model to implement countermeasures in the form of a feed-forward controller to overcome them and maintain accurate path following. This is accomplished through over 60 deployments into a variety of conditions to measure the wind and current phenomena and develop a model, based on our observations, of the effects of external forces on the ASV's behavior. Finally, the effect-based model feeds the improved autopilot controller to refine the ASV's navigation to overcome the currents and winds.

1.3 Contributions

The contribution of this work is the augmentation of the ASV's current control system with feed-forward controls to overcome the external dynamics and maintain a more accurate trajectory, by using measurements and models of natural disturbances affecting an ASV (Fig. 3) proposed in our previous work. Such a contribution will provide the greater scientific community with a more precise platform for data collection in challenging environments. In addition, it can provide an efficient and robust tool to aid search and rescue operators as well as environmental monitoring and bridge inspection teams.

The following section presents the methodology for accomplishing our goal as inexpensively as possible, with a brief discussion on the effects of external forces acting on an ASV, followed by a detailed proactive control augmentation description. Section 3 presents our experimental setup and approach to create a field trial testing environment to produce meaningful results in Sect. 4. Finally, we conclude with a short discussion of the results and suggestions for future work in this area.

2 Methodology

This section describes the strategy we employ to solve the problem presented in Sect. 1.1. For completeness, we will first briefly review the method for measuring external forces and modeling their effects on an ASV; for more information please

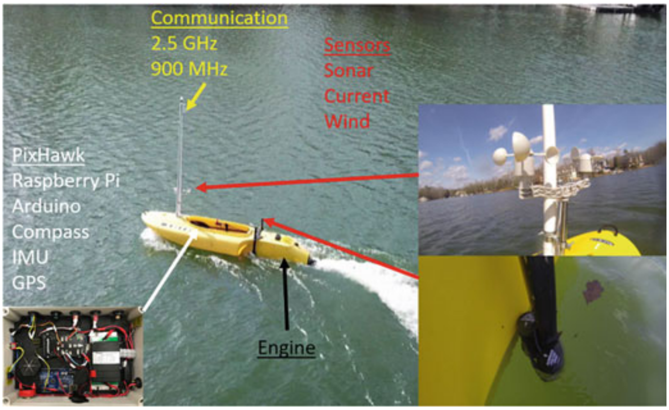


Fig. 3 UofSC's environmental dynamic measurement platform with anemometer, current sensors, depth sounder, GPS, IMU, robust communications, and a ROS-based data collection computer onboard

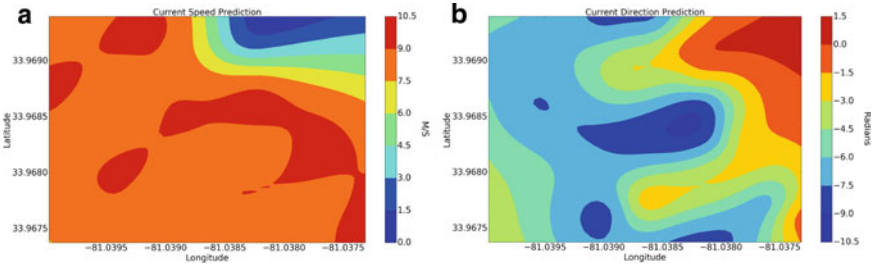


Fig. 4 **a** Current speed prediction, **b** current direction prediction during flood stage on Congaree River, SC

see Moulton et al. [12]. Then we present our approach of using these effects to implement proactive path-following control augmentation.

2.1 External Force Effects

There are two overlapping areas that benefit from measuring the external forces acting on the ASV. The first area, addressed in our prior work, is the ability to create a high-level force map of a given phenomenon (see Fig. 4). This capability enables planning algorithms such as the one proposed by Karapetyan et al. [6] to pre-select deployment sites and plan more efficient coverage solutions prior to launch. The second benefit results from the ability to use machine learning techniques for regression to produce effects models for the impact external forces are having on the robot. Figure 5 illustrates the impact vector that the external force is having on the

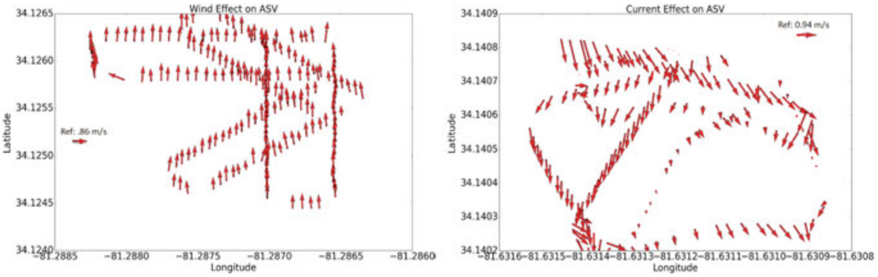


Fig. 5 The effects of the wind and current on the ASV. Illustration reflects different scales due to the dominant effect of current over wind on the ASV

ASV. This capability enables the work presented in this paper, which in high-level terms, the modeling of the effects feeds an adaptive controller which counteracts the external forces allowing for more accurate trajectory following of the ASV.

2.2 Proactive Control Through Way-Point Augmentation

Given an accurate model of the environment dynamics and an ability to predict temporally close external forces and their effects on the ASV, we seek to provide an augmentation to the Pixhawk way-point navigation controller. By manipulating the target global pose based on the measurements and effects of external forces we are able to provide intermediate way-points to the Pixhawk, coercing it to maintain the original desired trajectory, see Fig. 6. The intermediate way-points account for the effects of external forces and are calculated proportional to the distance d_t between the ASV and the goal way-point.

Pos_t is composed of the ASV's latitude, longitude, and velocity. F_t is comprised of the expected effect on the ASV's speed and heading resulting from the effects models in Sect. 2.1. Pos_n is the goal way-point and d_t is the distance between the ASV and Pos_n. Target'_n is a calculated intermediate way-point to send the controller to maintain the desired trajectory.

This portion of feed-forward augmented controller is illustrated in Fig. 7. The algorithm used to calculate the intermediate target way-points is presented in Algorithm 1.

The inputs to the algorithm are

- The measured current speed magnitude spd_c and direction dir_c,
- The measured wind speed magnitude spd_w and direction dir_w,
- The ASV position $(\text{lat}_t, \text{long}_t)$,
- The ASV speed spd_t and heading h_t,
- The target ASV speed spd_target,
- The list of way-points in the current mission.

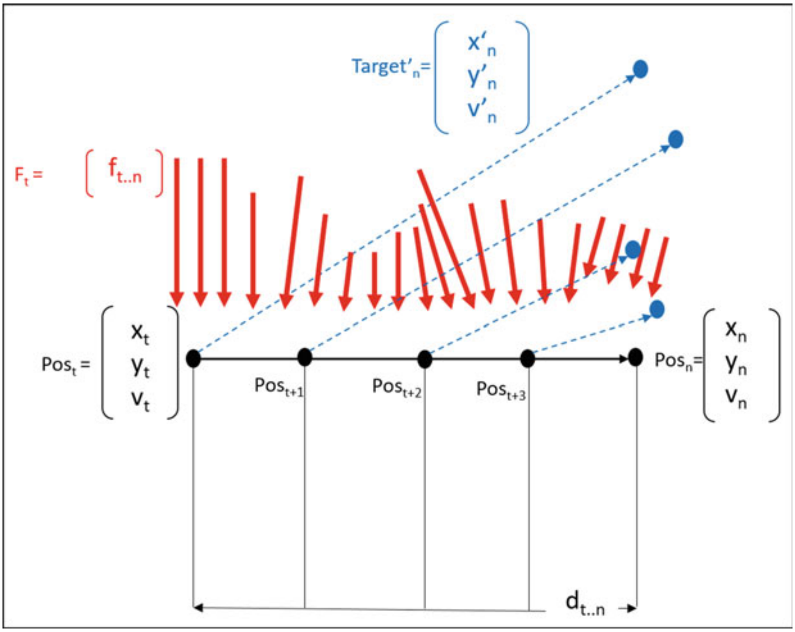

Target'$_n$= x'$_n$ y'$_n$ v'$_n$
F$_t$ = f$_{t..n}$
Pos$_t$ = x$_t$ y$_t$ v$_t$
Pos$_{t+1}$
Pos$_{t+2}$
Pos$_{t+3}$
Pos$_n$= x$_n$ y$_n$ v$_n$
d$_{t..n}$

Fig. 6 High-level illustration of way-point navigation augmentation method. Black solid line and position points denote the path we wish to maintain. Blue arrows represent the external force vector acting on the ASV, which are wind and current in our setup. Red points and arrows represent the intermediate way-points provided to the Pixhawk navigator and their associated target headings

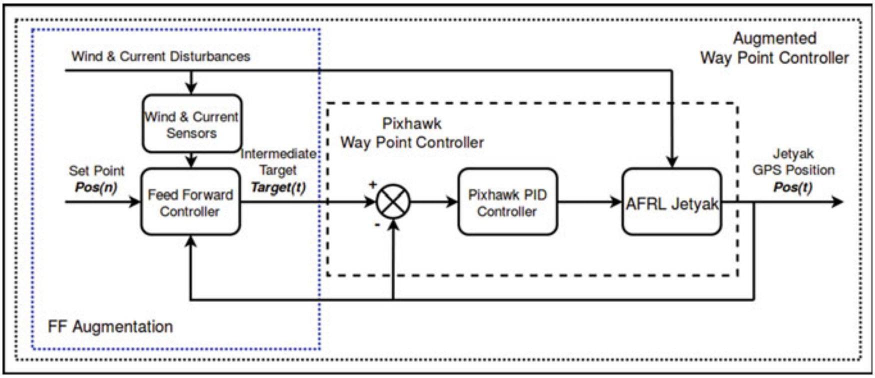

Augmented
Way Point Controller
Wind & Current Disturbances
Wind & Current Sensors
Set Point
Pos(n)
Feed Forward Controller
Intermediate Target
Target(t)
Pixhawk
Way Point Controller
+
-
Pixhawk PID Controller
AFRL Jetyak
Jetyak GPS Position
Pos(t)
FF Augmentation

Fig. 7 The way-point navigation PID controller used in the Pixhawk PX4 augmented by our intermediate way-point offset generator

The measurements are processed as they are received from the sensors during execution of each way-point from the mission. Based on the speed and orientation of the ASV, we determine the absolute values of each measurement and use that to predict with a linear regression the effect of the force on the speed and direction of the ASV (Line 7–8) [10]. While the target way-point is not reached, an intermediate way-point is calculated based on the effect_x and effect_y values. The speed is also adjusted based on the predicted error (Line 10). Finally, the ASV is sent to the newly calculated way-point (Line 11). When the new target position is processed by the Pixhawk navigation controller, it results in a smoother and more accurate path, and with this, we realize our original intended trajectory. In the following section, we will present the experiments carried out to demonstrate this capability.

Algorithm 1 Feed-forward Augmented Way-Point Navigation Controller

```
Input: $\text{spd}_c$, $\text{dir}_c$, $\text{spd}_w$, $\text{dir}_w$, ($\text{lat}_t$, $\text{long}_t$), $\text{spd}_t$, $h_t$, way-point list ($\text{lat}_n$, $\text{long}_n$), spd_target
Output: None
1: mission ← wp_list($\text{lat}_n$, $\text{long}_n$)
2: count ← $|mission|$
3: for each $i \in 1, \ldots,$count do
4:     go_to_waypoint($\text{lat}_i$, $\text{long}_i$, spd_target)
5:     wp ← $\text{lat}_i$, $\text{long}_i$
6:     while wp is not reached do
7:         effect_spd, effect_dir ← effect_model($\text{spd}_c$, $\text{dir}_c$, $\text{spd}_w$, $\text{dir}_w$, spd_target, $\text{spd}_t$, $h_t$)
8:         $\text{effect}_x$, $\text{effect}_y$ ← convert_to_coordinate_vectors(effect_spd, effect_dir)
9:         $\text{lat}'_i$, $\text{long}'_i$ ← calc_intermediate_wp($\text{lat}_i$, $\text{long}_i$, $\text{effect}_x$, $\text{effect}_y$)
10:        $\text{spd}'_i = \text{effect\_spd} + \text{spd\_target}$
11:        go_to_waypoint($\text{lat}'_i$, $\text{long}'_i$, $\text{spd}'_i$)
12:    end while
13: end for
```

3 Experiments

Over ten deployments were completed in support of this initiative, collecting and testing in over 190 km of river and lake environments; for testing the proposed controller, four of the deployments were in the river testing the control, while the rest established a baseline behavior and tested the effect of wind.

3.1 Platform

The base platform is a heavily modified Mokai Es-Kape,[1] termed Jetyak by UofSC's Autonomous Field Robotics Laboratory, shown in Fig. 3. The stock boat uses an internal combustion Subaru engine and is capable of speeds up to 22.5 km/h and a deployment duration of over eight hours with reduced speed. The ES-Kape's factory pulse width modulated controlled servo system makes it ideal for robotic control integration.

On top of the stock configuration, this research vessel is outfitted with a Pixhawk[2] flight control system. Outfitted with remote control from Taranis, long-range mission tracking through RFD 900+ radios, and on-board control through a companion Raspberry Pi serving to host a Robot Operating System (ROS) [14] node, the Jetyak maintains robust architecture for autonomous initiatives as well as redundant override capabilities to ensure safe testing and experimentation.

The sensing capability is provided through NMEA 0183 depth sonar, Sparkfun anemometer for wind, and Ray Marine ST 800 paddle wheel speed sensors for current measurements. Current and wind sensors require analog to digital drivers due to the inexpensive sensing route we have selected. This is provided through Arduino Mega and Weathershield micro-controllers, respectively.

3.2 Experimental Approach

In order to provide experimental results that are easily comparable to the original way-point PID controller, we use straight-line test trajectories that run in the cardinal directions parallel and perpendicular to the predominant external force. In this case, currents are being tested, and we illustrated in Fig. 2 that the Jetyak's poorest path-following performance occurs when traveling in the same direction as the current. This led us to select the four cardinal and four intermediate direction orientations to the current as our test baseline, shown in Fig. 8. Straight-line segments were produced to replicate the most common patterns from route planning experiments. The generated segments were initially used as input to the standard Pixhawk way-point controller. Then, the same segments were used as input to the augmented controller with the intermediate way-points enabled.

Given this controlled experimental setup, the results in Sect. 4 illustrate the success of this approach as well as directions for future work.

---

[1] http://www.mokai.com/mokai-es-kape/.

[2] https://docs.px4.io/en/flight_controller/mro_pixhawk.html.

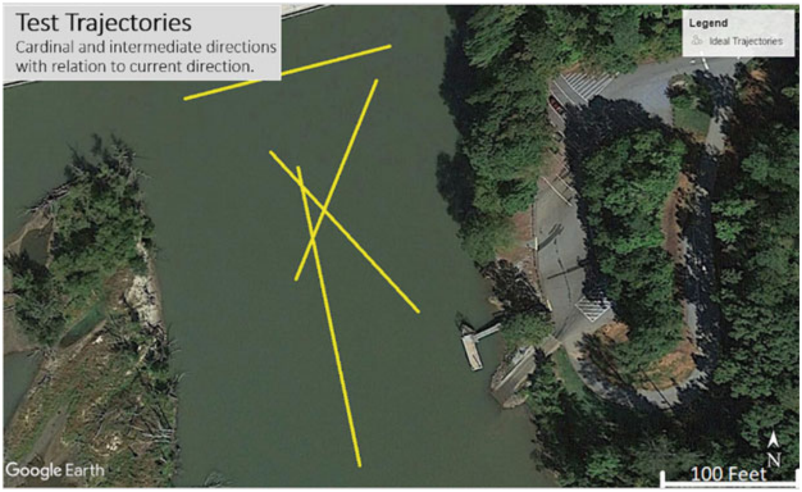

Test Trajectories
Cardinal and intermediate directions with relation to current direction.
Legend
Ideal Trajectories
Google Earth
100 Feet
N

Fig. 8 Test patterns run in both directions to establish a control baseline for performance evaluation in currents of less than 1 m/s, depending on location of the ASV in the Saluda River's cross-section

4 Results

In this section, we will compare the performance of the standard Pixhawk GPS way-point navigation controller with and without the proposed feed-forward augmentation. Since the standard controller performs well in upstream maneuvers, the focus will be on the performance difference in the downstream cases. Due to weather constraints during the field trials, the results presented in this paper were obtained from data collected in a river with an average measured speed of 0.677 m/s during the trial for straight trajectories.

4.1 Way-Point Navigation

As illustrated in Fig. 9, the built-in Pixhawk controller is generally able to reach the required way-points. However, the PID coefficients are tuned to operate in a specific environment. When changing environments, the PID coefficients should be tuned again. This task becomes insurmountable when operating in environments with ever-changing dynamic forces at play. As shown in Fig. 9 left, negotiating currents in upstream to perpendicular directions is relatively stable. This is due to the fact that the speed of the ASV relative to the ground is slightly reduced, allowing enough time for the PID controller to compensate for the error. However, in Fig. 9 right, we see the opposite effect when the speed of the ASV relative to the ground is increased, thereby accumulating too much error in the PID controller to overcome the external forces. This typically results in an overshoot scenario where the ASV begins harmonically

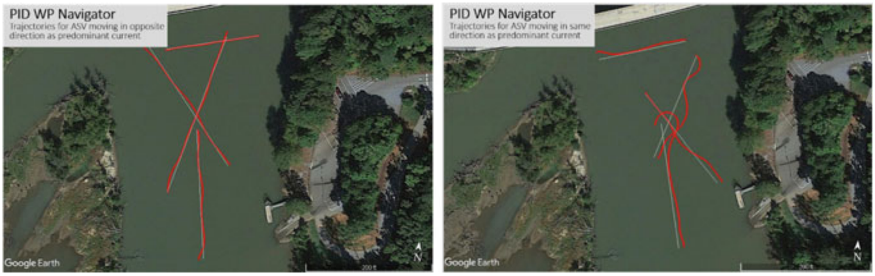

PID WP Navigator
Trajectories for ASV moving in opposite direction as predominant current
Google Earth
N
PID WP Navigator
Trajectories for ASV moving in same direction as predominant current
Google Earth
N

Fig. 9 Pixhawk PID controlled way-point navigator tracking in slow currents with the ASV travelling mainly **a** against the predominant direction of the current; **b** with the predominant direction of the current—white line: target trajectory, red line: actual executed trajectory

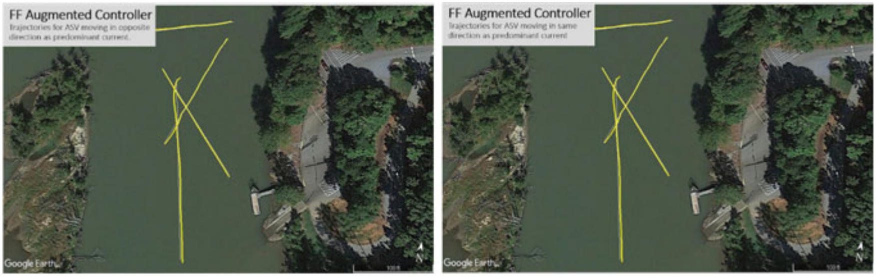

FF Augmented Controller
Trajectories for ASV moving in opposite direction as predominant current
Google Earth
N
FF Augmented Controller
Trajectories for ASV moving in same direction as predominant current
Google Earth
N

Fig. 10 Augmented Pixhawk way-point navigator tracking in slow currents with the ASV travelling mainly **a** against the predominant direction of the current; **b** with the predominant direction of the current—white line: target trajectory, yellow line: actual executed trajectory

oscillating back and forth over the desired trajectory. It should also be noted, that as the speed of the current increases, this behavior starts to present itself in trajectories perpendicular to the current. Adjusting the integral gain in the PID controller can help solve this problem, but it will also produce undesirable oscillatory behavior in upstream trajectories.

4.2 *Proactive Effects Augmented Way-Point Navigation Controller*

As illustrated in Fig. 10, by augmenting the built-in PID controller in the Pixhawk, we were able to follow much more precisely the desired path to each way-point than the non-augmented controller. These results serve as proof of concept for Algorithm 1. As shown in Fig. 10, path following in currents in all orientations to the ASV is qualitatively improved.

The results in Table 1 show a quantitative comparison of the performance of our augmented proactive controller with the baseline way-point navigator. In particular, a

Table 1 Comparing the performance of the standard Pixhawk way-point PID controller with our intermediate way-point augmented control. Perpendicular results represent average error over two traversals in opposing directions

	ASV trajectory relative to current							
		Perpendicular	Parallel with	Parallel against	L-R diagonal with	L-R diagonal against	R-L Diagonal With	R-L Diagonal against
WP navigator with PID control	Max Error	4.50 m	9.32 m	3.86 m	3.46 m	1.63 m	7.85 m	2.57 m
	% Path Error >1 m (%)	44.2	76.8	18.3	48.6	12.7	83.5	40.1
Augmented WP navigator with PID control	Max Error	1.58 m	1.48 m	1.08 m	0.75 m	0.74 m	1.07 m	0.68 m
	% Path Error >1 m (%)	9.3	11.9	7.9	0	0	6.3	0

marked improvement can be observed in both maximum error and percentage of the path that is more than a meter far from the target trajectory. Max error represents the largest distance between the straight-line trajectory and the actual path of the ASV. Percentage path error greater than one meter quantifies the portion of the path where the ASV was more than one meter from the ideal trajectory. Confirming intuition, the ability of the augmented control algorithm to change the forward thrust of the ASV provides the largest numerical improvement when moving with the predominant direction of the current.

5 Conclusions

The path-following precision achieved by this work can have profound impacts for the research, emergency services, and exploration communities. The ability to provide bathymetric surveying and mapping capabilities to remote areas with highly dynamic currents will enable researchers to expand the boundary between known and unknown environments.

To improve the robustness of the control augmentation presented, future work should include two areas. First, the addition of providing the same precision path following for a Dubin's vehicle, such as the Jetyak will require additional methods to handle deliberate turns in planned missions. Second, another desirable expansion of this work will include changing from intermediate way-point augmentation to a lower level control of the linear and angular velocities (v, ω). Such an approach may produce more concise countermeasures to further reduce the path tracking error.

Acknowledgements The authors would like to thank the generous support of the National Science Foundation grants (NSF 1513203, 1637876).

References

1. Curcio, J., Leonard, J., Patrikalakis, A.: SCOUT–a low cost autonomous surface platform for research in cooperative autonomy. In: OCEANS MTS/IEEE, pp. 725–729. IEEE (2005)
2. Fraga, J., Sousa, J., Cabrita, G., Coimbra, P., Marques, L.: Squirtle: an ASV for inland water environmental monitoring. In: ROBOT2013: First Iberian Robotics Conference, pp. 33–39. Springer (2014)
3. Girdhar, Y., Xu, A., Dey, B.B., Meghjani, M., Shkurti, F., Rekleitis, I., Dudek, G.: MARE: marine autonomous robotic explorer. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5048–5053 (2011)
4. Hsieh, M.A., Hajieghrary, H., Kularatne, D., Heckman, C.R., Forgoston, E., Schwartz, I.B., Yecko, P.A.: Small and adrift with self-control: using the environment to improve autonomy. Robotics Research, pp. 387–402. Springer (2018)
5. Karapetyan, N., Braude, A., Moulton, J., Burstein, J.A., Whitea, S., O'Kane, J.M., Rekleitis, I.: Riverine coverage with an autonomous surface vehicle over known environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2019), *accepted*

6. Karapetyan, N., Moulton, J., Lewis, J.S., Quattrini Li, A., O'Kane, J.M., Rekleitis, I.: Multi-robot Dubins coverage with autonomous surface vehicles. In: IEEE International Conference on Robotics and Automation (ICRA) (2018)
7. Kimball, P., Bailey, J., Das, S., Geyer, R., Harrison, T., Kunz, C., Manganini, K., Mankoff, K., Samuelson, K., Sayre-McCord, T., Straneo, F., Traykovski, P., Singh, H.: The WHOI Jetyak: an autonomous surface vehicle for oceanographic research in shallow or dangerous waters. In: IEEE/OES Autonomous Underwater Vehicles (AUV), pp. 1–7 (2014). https://doi.org/10.1109/AUV.2014.7054430
8. Lee, H.S., Tomizuka, M.: Robust motion controller design for high-accuracy positioning systems **43**(1), 48–55 (1996)
9. Mahacek, P., Berk, T., Casanova, A., Kitts, C., Kirkwood, W., Wheat, G.: Development and initial testing of a swath boat for shallow-water bathymetry. In: OCEANS 2008, pp. 1–6. IEEE (2008)
10. Moulton, J.: A novel and inexpensive solution to build autonomous surface vehicles capable of negotiating highly disturbed environments. Ph.D. thesis, University of South Carolina (2019)
11. Moulton, J., Karapetyan, N., Bukhsbaum, S., McKinney, C., Malebary, S., Sophocleous, G., Quattrini Li, A., Rekleitis, I.: An autonomous surface vehicle for long term operations. In: OCEANS 2018 MTS/IEEE Charleston, pp. 1–10. IEEE (2018)
12. Moulton, J., Karapetyan, N., Quattrini Li, A., Rekleitis, I.: External force field modeling for autonomous surface vehicles. arXiv preprint arXiv:1809.02958 (2018)
13. Pereira, A., Das, J., Sukhatme, G.S.: An experimental study of station keeping on an underactuated asv. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3164–3171. IEEE (2008)
14. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5. Kobe, Japan (2009)
15. Rasal, K.: Navigation & control of an automated swath surface vessel for bathymetric mapping. Santa Clara University (2013). https://scholarcommons.scu.edu/mech_mstr/2. Accessed 18 Oct 2017
16. Rodriquez, D., Franklin, M., Byrne, C.: A study of the feasibility of autonomous surface vehicles. Worcester Polytechnic Institute (2012). https://web.wpi.edu/Pubs/E-project/Available/E-project-121212-135500/unrestricted/ASV_IQP.pdf
17. Tsao, T.C.: Optimal feed-forward digital tracking controller design. J. Dyn. Syst. Meas. Contr. **116**(4), 583–592 (1994)

Airborne Particle Classification in LiDAR Point Clouds Using Deep Learning

Leo Stanislas, Julian Nubert, Daniel Dugas, Julia Nitsch, Niko Sünderhauf, Roland Siegwart, Cesar Cadena, and Thierry Peynot

1 Introduction

LiDAR sensors are used in many areas of robotics such as obstacle detection [2, 10], localization [3, 9], and semantic mapping [7, 8]. In cases of weak or changing illumination or when high spatial resolution is needed, LiDARs often have an advantage compared to RADAR sensors or cameras [11]. However, LiDAR sensors are sensitive to particles in the air such as dust, fog, or smoke [13, 14]. This impacts a given point cloud as depicted in Fig. 1. Present-day algorithms are often suffering from corruptions of this nature. For instance, *Boss*—the winner of the DARPA Urban Challenge - was led to a (temporary) stop because a dust cloud in the LiDAR data was perceived as an obstacle [24][1]. This work aims to detect LiDAR points generated by airborne particles to reduce their impact on robotics algorithms.

Limited work has been done in this field. Gerardo et al. [6] used a RADAR and LiDAR to perform sensor data fusion and discarded the LiDAR data whenever there was an inconsistency in the data between the sensors, exploiting the fact that RADAR data is not affected by airborne particles. Peynot and Kassir [12] used a similar approach by detecting discrepancies between the LiDAR range and camera intensity

---

[1] https://leo-stan.github.io/particles_detection_fsr

Leo Stanislas and Julian Nubert are equaly contributors

J. Nubert · D. Dugas · R. Siegwart · C. Cadena
ETH Zurich, Zurich, Switzerland
e-mail: nubertj@ethz.ch

J. Nitsch
Ibeo Automotive Systems, Hamburg, Germany

L. Stanislas (✉) · N. Sünderhauf · T. Peynot
Queensland University of Technology, Brisbane, Australia
e-mail: l.stanislas@qut.edu.au

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_28

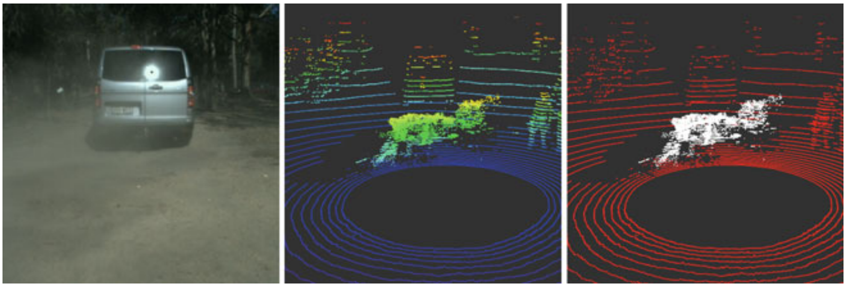


Fig. 1 Left: Image of an experimental scene with dust behind a car. Middle: LiDAR point cloud with dust corruption (colored by height). Right: Predicted particles (white) and non-particles (red)

gradients. However, both of these approaches require multiple types of sensors. A classification approach [20] was proposed using a LiDAR sensor, by predicting individual LiDAR points generated by airborne particles; however, it was only tested on traditional classifiers (*Support Vector Machine* and *K-Nearest Neighbor*). In our previous work [21], we pursued a similar approach to [20] but evaluating more modern classifiers. We obtained promising results and showed that a voxel-based deep learning classifier outperformed traditional classifiers such as *Random Forest* or *Support Vector Machine*.

This paper expands on the LiDAR-based classification approach. Since the impact on LiDAR data across different types of particles is comparable, we do not make the distinction between types of airborne particle and define two classes for each LiDAR point: *particle* (e.g., dust or smoke) or *non-particle* (e.g., solid object). We present and compare two deep-learning classification approaches:

Voxel-wise classification: Each point cloud is discretized into 3D voxels. A Neural Network architecture, significantly improved from our previous work [21], is used to perform classification on a voxel level.

Point-wise classification: Each point cloud is formatted into a 2D LiDAR image representation with each pixel corresponding to an individual LiDAR return. A *Convolutional Neural Network (CNN)* based on the U-Net architecture [18] performs point-wise classification on the 2D LiDAR image.

We explore different classification features extracted from the LiDAR data including geometry, intensity, and multi-echo information and evaluate the best combination of these features for each classification approach. Additionally, we give insights into the ability of our classifiers to distinguish between different types of particles for applications such as autonomous driving where decisions depend on environmental conditions (e.g., snow or fog). We evaluate our work on a realistic outdoor dataset with the presence of fog and dust particles in LiDAR point clouds. The dataset is labeled semi-automatically using a background subtraction method based on occupancy grids, which allowed for increasing the amount of labeled data from our previous work [21] by a factor of ten. The two approaches presented in this work significantly outperform the state of the art in the classification of airborne particles in LiDAR point clouds. This is also the first demonstration of a benefit in using LiDAR multi-echo returns as a classification feature. Additionally, our point-wise classification approach is a novel use of a *CNN* on 2D LiDAR images to detect airborne particles. Finally, we demonstrate the significance of this work on two robotics use cases; a relative pose estimation task and an obstacle detection task, both in the presence of airborne particles. We show a clear performance increase in each task thanks to our approaches.

The remainder of this paper is structured as follows. In Sect. 2, we discuss the related work. Sect. 3 introduces the methodologies of our classification approaches. In Sect. 4, we provide details on our experimental setup while in Sect. 5 we present our experimental results. Finally, in Sect. 6 we conclude and consider future work.

2 Related Work

The choice of input features is an important part of any classification approach and LiDAR data contains various types of information. The geometry information is contained in the 3D position of each LiDAR point. Lalonde et al. [8] used the geometry information in the form of a *Principal Component Analysis (PCA)* to differentiate ground surfaces from non-ground surfaces and unstructured elements. The 3D position information of each LiDAR point was also used in [27] as part of a deep learning approach. The intensity (also referred to as remittance) information is partly correlated to the type of material hit by a LiDAR return. This was used in multiple terrain classification applications [23, 25] as well as for object detection [1, 27]. Finally, most modern LiDAR sensors are capable of returning multiple echos for each LiDAR ray (multi-echo). This was used as a classification feature in [17] in a terrain classification task. However, the authors did not find this feature to be beneficial due to the low proportion of meaningful multi-echo returns in their experiments. In our previous work [21], we used a combination of geometry and intensity information to detect airborne particles in LiDAR point clouds. In this work, we investigate various combinations of geometry, intensity and multi-echo information.

Multiple deep learning approaches have been used to classify LiDAR point clouds. A voxel-based approach was developed by Zhou et al. [27], achieving state-of-the-art performance on the object detection task in the KITTI dataset [5]. 3D voxels are used to aggregate classification features from LiDAR points. Using voxels provides the ability to change the resolution of the data representation. This approach was adapted in our previous work [21] to perform the voxel-wise classification of airborne particles in LiDAR point clouds. While the results were promising, the classifier could not exploit the geometry information. In this work, we improve this architecture by adding three sequential 3D convolutions to learn the spatial structure of the data. Alternatively, Qi et al. developed a point-based classification approach on unordered LiDAR point clouds for semantic segmentation of indoor scenes [15]. However, this approach does not make use of the local structure of the data. Although this shortcoming was later addressed with a hierarchical recursive approach [16], this increased the computational cost of inference. Another point-based classification approach was proposed by Zhang et al. [26] using a *Convolutional Neural Network (CNN)* based on the *U-Net* architecture [18] on a 2D image representation of the LiDAR point clouds. Their image representation was based on a bird's eye view of a 3D voxel grid with the vertical axis as the feature channel. This approach outperformed the aforementioned point-based approach from Qi et al. on semantic segmentation tasks in two different datasets. In this work, we also use the *U-Net* architecture to perform point-wise classification; however, our LiDAR image representation is a 2D projection of the 3D point cloud into a 2D image where each pixel corresponds to a LiDAR return. This representation is advantageous with constant size and density regardless of the observed scene.

3 Airborne Particle Classification in LiDAR Point Clouds

We structure our binary classification approach as a four-step process illustrated in Fig. 2, taking a LiDAR point cloud as input and returning the same point cloud with a class label associated to each point. First, the features are selected from the point cloud information returned by the sensor, followed by a formatting step to prepare the data for a particular representation depending on the classification approach. The neural network then predicts the class of the input. Finally, a post-processing step is applied to convert the data from the prediction representation back to the original point cloud format with predicted labels added.

3.1 Classification Input Features

One of the main objectives of this work is to evaluate the best combination of LiDAR-based features for the classification of airborne particles. We investigate three different types of features:

Geometry: With the high density of the point clouds provided by modern 3D LiDAR sensors, the geometry (i.e., 3D position of points) provides information on the shape of elements in the scene. This information is especially useful when using convolu-

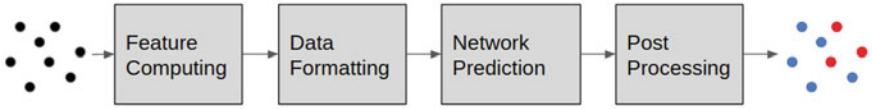


Fig. 2 The four stages of the point cloud classification process

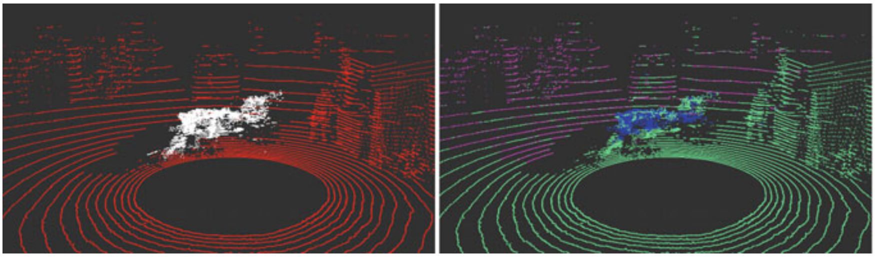


Fig. 3 LiDAR point cloud from the example scene from Fig. 1. Left: Colored by label with *particles* in white and *non-particles* in red. Right: Colored by echo feature with green when both echoes match, blue for the first echo return, and purple for the second echo return

tions in a neural network architecture. Airborne particles will generally produce an unstructured cloud while other elements will be more structured (e.g., walls, cars).

Intensity: Intensity values depend on the amount of light measured by the LiDAR sensor. Airborne particles tend to return very low-intensity values due to their small size, scattering most of the light emitted by the sensor. This helps to differentiate particles from surfaces made of a material with higher reflectivity such as grass, concrete, and wood.

Multi-Echo: Different echoes can be measured when multiple peaks are detected in the light intensity of a single light ray at distinct ranges. This phenomenon is common for airborne particles where a first echo can be returned from a cloud of particles, while a subsequent one can go through that cloud and hit an object behind it [14]. Figure 3 shows that most *non-particle* elements (red) have both echoes matching (green) whereas *particles* (white) tend to return a first echo (blue) while the obstacles behind return a second echo (purple).

3.2 Voxel Classification Approach

Representing a LiDAR point cloud with voxels enables various levels of abstraction in the data, as well as concatenating features between neighboring points. This approach is an incremental improvement of the voxel-based approach presented in our previous work [21].

Data Representation: We discretize each LiDAR point cloud into equally spaced non-overlapping 3D voxels of equal size $[v_X, v_Y, v_Z]$. We only create a voxel if there is at least one LiDAR point inside it. This forms a sparse voxel grid containing N voxels for a given LiDAR point cloud, with a maximum number of voxels in each dimension defined by $[W, H, D]$. The voxel grid is centered around the LiDAR sensor. Each voxel contains a maximum of T LiDAR points $p = [x, y, z, a, e]$ with x, y, z being the absolute position, a the intensity with values in the range [0; 100],

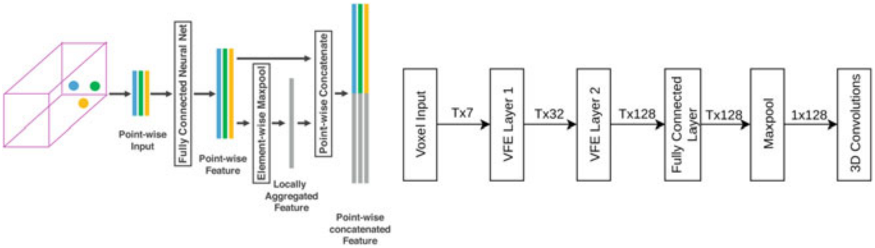

Point-wise Input
Fully Connected Neural Net
Point-wise Feature
Element-wise Maxpool
Locally Aggregated Feature
Point-wise Concatenate
Point-wise concatenated Feature
Voxel Input
Tx7
VFE Layer 1
Tx32
VFE Layer 2
Tx128
Fully Connected Layer
Tx128
Maxpool
1x128
3D Convolutions

Fig. 4 Left: Voxel Feature Encoding (VFE) layer [27]. Right: Voxel classification architecture

and e the echo status with three possible values: 0 when both strongest and last echoes match, and when the echoes are different: 1 when the echo is the strongest and 2 when it is the last echo. The input feature vector of our voxel classification architecture is defined as

$$\mathbf{F_{in}} = \{\{[x_i - \hat{x}, y_i - \hat{y}, z_i - \hat{z}, a_i, e_{0,i}, e_{1,i}, e_{2,i}]\}_{i=1...T_n}\}_{n=1...N}, \tag{1}$$

where $\hat{x}, \hat{y}, \hat{z}$ is the average of the position of all points in the voxel. $x_i - \hat{x}, y_i - \hat{y}, z_i - \hat{z}$ is the position of each LiDAR point relative to the other points in the same voxel. a_i is the intensity return, and $e_{0,i}, e_{1,i}, e_{2,i}$ are the one-hot encoded values corresponding to the echo value each taking the binary value 0 or 1. This makes our input feature tensor F_{in} of shape $[N, T, 7]$ corresponding to one scan of the LiDAR sensor where N (the number of voxels created from the scan) is used as batch size. Finally, the predicted label of each voxel is assigned to all LiDAR points inside to provide a label per LiDAR point for this evaluation.

Network Architecture: Our network architecture is based on the *VoxelNet* architecture from Zhou et al. [27] and is depicted in Fig. 4. Each voxel in the input tensor F_{in} is passed to two *Voxel Feature Encoding (VFE)* modules and a fully connected layer (see Fig. 4) to learn features within each voxel. We apply a *maxpool* operation on the output of the fully connected layer to concatenate the features over all T points. Batch-normalization and *Relu*-activation is used after each fully connected layer inside the *VFE* modules. We apply 3D convolution to the sparse tensor of shape $128 \times W \times H \times D$ formed with the computed features of each voxel. We apply three 3D convolutions sequentially with *Conv1*(128,64,k,s,p), *Conv2*(64,64,k,s,p), and *Conv3*(64,2,k,s,p), where k=(1,1,1) is the kernel size, s=(1,1,1) is the stride, and p=(1,1,1) is the padding. The output of the network is a sparse tensor of shape $2 \times W \times H \times D$ to which we apply a *Softmax* operation on the prediction score. The predicted label for all N voxels in each scan is obtained from this output. We apply a standard scaling operation by removing the mean and scaling to unit variance on the input values of each voxel. We reduce the imbalance in the training data by using a small voxel map around the zone with particles with $W = H = 100$ and $D = 15$ and a voxel size of $v_X = v_Y = v_Z = 0.2m$ during training.

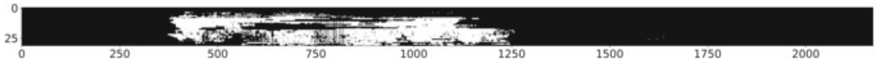


Fig. 5 Mask of LiDAR image (class for each of the pixels). White: *particle*, black: *non-particle*

3.3 Point Classification Approach

Data Representation: We render each 3D LiDAR scan in a 2D LiDAR image with each pixel corresponding to an emitted ray and the rows and columns of this image correspond to the vertical and horizontal angular resolution of the LiDAR sensor respectively. This is possible thanks to the inherent LiDAR property of sequentially scanning the environment. Using the angular resolution of the 3D LiDAR sensor, an image-width C can be computed while the height R is dependant on the number of vertical rings of the LiDAR sensor. The horizontal pixel coordinate is obtained by mapping the polar angle of the ring to the image space. Each pixel contains up to four channels corresponding to the range and intensity of both echo returns for a given LiDAR beam. We define this LiDAR image representation as

$$\mathbf{F_{in}} = \{p_{ij} = [r_{echo1,ij}, a_{echo1,ij}, r_{echo2,ij}, a_{echo2,ij}]\}_{i=1...R, j=1...C}, \tag{2}$$

where $\mathbf{F_{in}}$ is the LiDAR image used as input tensor to the network, p_{ij} is the pixel corresponding to the vertical angle index i and horizontal angle index j. $r_{echo1,ij}$ and $a_{echo1,ij}$ are the range and intensity values for the first echo, while $r_{echo2,ij}$ and $a_{echo2,ij}$ are the range and intensity values for the last echo. R and C are the number of vertical and horizontal angle increments, respectively. An example of the mask for the presented representation as an outcome of the labeling can be seen in Fig. 5.

Network Architecture: Due to spatial correlations of neighboring pixels, we use the *U-Net* architecture [18] to perform 2D convolutions on the input data. Figure 6 depicts the structure of this network. *Maxpool* layers are used in the encoder to create smaller features with high expressiveness. In the decoder, the network extracts information from these meaningful features to classify each of the pixels. *Maxpool* and *up-sampling* operations are only performed along the input columns whereas convolutions are applied on both rows and columns with 3 × 3 filters.

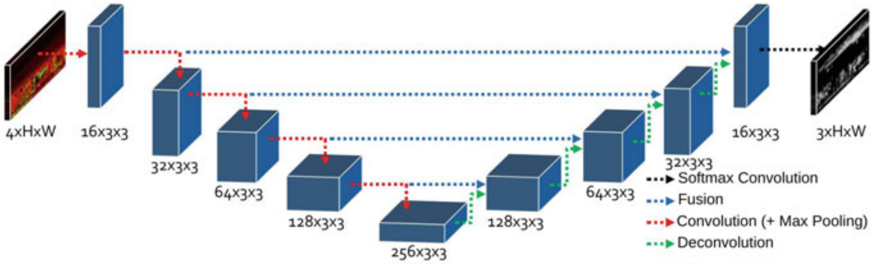


Fig. 6 The U-Net Architecture used for point classification

By having four pooling layers, the horizontal resolution is reduced by a factor of $2^4 = 16$ in the encoder. Therefore, the width of the input LiDAR image has to be a multiple of 16 for the output to have the same shape as the input. Since the network is *fully convolutional*, it is able to process inputs of variable sizes. To train the network, we take a sample of size $R = 32$ and $C = 512$ as an input vector to the network chosen randomly from the original LiDAR image. Additionally, the image is horizontally flipped with a probability of 50%. These two data augmentation methods prevent the network from overfitting the training data. To mitigate dataset imbalance, we only use frames with a high ratio of particles for training and a combination of the *binary cross entropy* and the *Dice Loss* [22] as cost function. The latter has shown advantages for training neural networks on imbalanced data.

4 Experimental Setup

4.1 Dataset

To evaluate our methods, we use the dataset from our previous work [21] containing dust and fog particles in 19 different outdoor scenes with elements such as cars, humans, buildings, shrubs, and trees. The sensor data were collected using a Velodyne HDL-32E LiDAR sensor mounted on a Clearpath Husky platform and contains position, intensity, and multi-echo information in the point clouds. Each scene was recorded statically with and without particles to allow for semi-automated labeling using background subtraction. The noise in the LiDAR point 3D position makes a simple point-to-point background subtraction inappropriate. Therefore, we used an occupancy grid representation where a particle-free reference grid was subtracted to the grid generated for each scan to be labeled. We use a Polar representation with logarithmic resolution on the radial axis for the occupancy grid to accommodate for the circular structure of the LiDAR scans and the lowering point density in the far field. This labeling implementation is included with the code for this work. Overall, our labeled dataset contains 16,247 fully labeled LiDAR scans, which is an increase by a factor of ten from our previous work. Images of the scenes in the dataset are provided with the code for this work.

To evaluate our approaches, we used nine scenes as a training set, two scenes as a validation set, and the remaining eight scenes as a test set. We made sure the scenes were sufficiently different configurations from training to testing to prevent overfitting. All results were computed on the entire test set unless specified otherwise.

Table 1 Parameters configurations for both classification approaches

	Voxel classification	Point classification
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Optimizer parameters	$\beta_{1/2}$ = (0.9, 0.9), ϵ = 10^{-08}, weight decay = 0	$\beta_{1/2}$ = (0.9, 0.999), ϵ = 0, weight decay = 0
Training	10,000 iterations, batch size = N	271,150 iterations, batch size = 32
Loss function	Binary cross entropy	Combination of *BCE* and *DL*
Trainable parameters	355,052	1,857,859
Input size	$T = 35$	$R = 32$, $C_{pred} = 2160$, $C_{train} = 512$

4.2 Performance Metrics

We evaluate the performance as a binary classification problem with a positive prediction (P) corresponding to a *particle* and a negative prediction (N) corresponding to a *non-particle*. We compute the *precision* and *recall* scores as $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$. TP and TN denote the number of true positives and true negative predictions, respectively, and FP and FN the number of false positive and false negative predictions. Finally, we compute the F1 score as $F1 = 2\frac{precision \cdot recall}{precision + recall}$.

4.3 Evaluation Parameters

We evaluate the voxel-based approach with a map size of $W = H = 400$ and $D = 15$ and a voxel size of $v_X = v_Y = v_Z = 0.2m$. An overview of the parameters for both neural network architectures are noted in Table 1.

5 Experimental Results

Each evaluation is computed by training multiple networks on our training and validation sets and averaging their performance on the prediction of our test set. This provides a more meaningful performance interpretation by reducing the influence of statistical training outliers.

5.1 Voxel Classification Evaluation

We evaluate the performance of this approach for six different configurations of input features. For each combination, we train five models and average their performance on our test set. The results are presented in Table 2. We compare the performance to the voxel classification approach from our previous work evaluated on the dataset used in this paper. Most combinations of features outperform the architecture from our previous work. The intensity and geometry features perform better on their own than combined. Adding multi-echo to the intensity feature decreases performance; however, it increases performance when added to the geometry feature. Finally, this approach performs best by combining all three features achieving a maximum F1 score of 91%.

5.2 Point Classification Evaluation

The averaged performance of our point classification approach for different input feature combinations is presented in Table 3. The multi-echo feature improves performance when combined with geometry. The geometry is needed to make the best of this architecture. We obtain the highest scores for our point classification approach using *Geometry* and *Multi-Echo* achieving an F1 score of 94%. We observe F1 scores of more than 85% for using intensity values only.

Table 2 Averaged voxel classification performance for all six input configurations and our previous work

	Previous work [21]	Geometry	Intensity	Geometry intensity	Geometry multi-echo	Intensity multi-echo	Geometry intensity multi-echo
Precision	0.37	0.53	0.64	0.42	0.77	0.31	**0.89**
Recall	**0.97**	0.90	0.91	**0.97**	0.95	**0.97**	0.93
F1	0.54	0.65	0.75	0.58	0.85	0.44	**0.91**

Table 3 Averaged point classification performance for all six input configurations

	Geometry	Intensity	Geometry intensity	Geometry multi-echo	Intensity multi-echo	Geometry intensity multi-echo
Precision	0.65	0.87	0.87	**0.94**	0.82	0.92
Recall	0.96	0.90	**0.96**	0.95	0.93	**0.96**
F1	0.73	0.89	0.91	**0.94**	0.87	0.93

Table 4 Comparison of classification results between our approaches with the best configurations of input features and the state of the art

	Previous work [21]	Voxel classification	Point classification
Precision	0.37	0.89	**0.94**
Recall	**0.97**	0.93	0.95
F1 score	0.54	0.91	**0.94**

5.3 Performance Comparison

In Table 4, we present the classification performance using the best input feature combination for each approach and compare them to the state of the art. The performance of the voxel-based classification and the point-based classification is comparable while both are clearly outperforming the state of the art. The input features used to obtain these results will be used in the rest of this paper.

5.4 Differentiate Dust from Fog Particles

For some applications, it is beneficial to differentiate between different types of particles. For instance, firefighters might only be interested in smoke clouds when assessing a bush fire while removing other types of particles. We evaluate whether our approaches are able to differentiate the type of particle by defining three classes *dust*, *fog*, and *non-particle*. Our voxel-based approach was not able to successfully learn the difference between the fog and dust particles. This is likely due to the simpler structure of the network. However, our point classification architecture was successfully detecting the difference between the two types of particles with F1 scores of 96% for the *fog* class and 92% for the *dust* class.

5.5 Use Cases

5.5.1 Pose Estimation Task

LiDAR sensors are widely used for robot localization tasks [3]. A well-known approach to perform pose estimation using LiDAR data is using Point Cloud Matching to estimate the transformation between two point clouds taken from different positions. However, this matching process can be altered by the presence of moving airborne particles in the point cloud. We demonstrate the significance of our work by showing a decrease in pose estimation error and faster convergence when removing

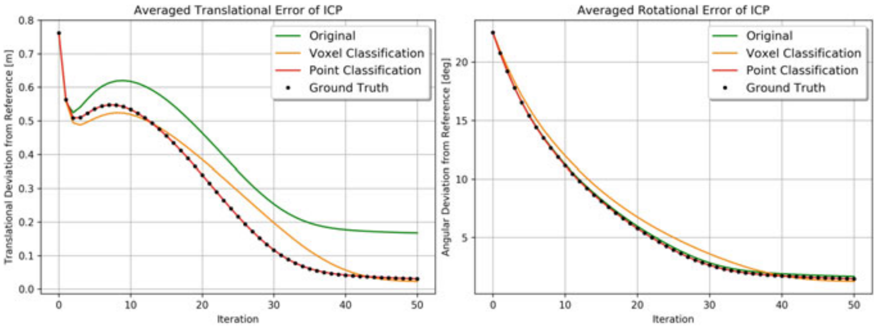


Fig. 7 Left: averaged translational error. Right: Averaged rotational error. Different configurations: original point cloud, and filtered point cloud using ground truth, voxel classification, point classification

airborne particles detected in LiDAR point clouds prior to performing the matching step using a point-to-point *ICP* algorithm [19].

We introduce an erroneous translation and rotation to our statically recorded dataset to produce corrupted scans and compare them to a reference scan of the scene free of particles. We apply the ICP algorithm to match the corrupted scan to the reference and measure the remaining translation and rotation error. The error introduced in each scan is a random rotation in the interval of $[0°, 45°]$ around the vertical axis followed by random translations in the interval of $[0m, 1m]$ along the vertical and horizontal axes. We perform this evaluation on 10% of the total scans in our test set and compute the average error in translation and rotation. This evaluation is performed on the original point cloud containing all particles and three filtered point clouds for which particles have been removed according to (1) the ground truth, (2) the point classification approach, and (3) the voxel classification approach. The ground truth was obtained by performing ICP on a point cloud from which particles have been removed (filtered) using the ground truth labeled data. The results are presented in Fig. 7 where the ideal value is a nil translation and rotation since the sensor is static between scans. The impact of airborne particles in LiDAR point clouds on the translational matching of the ICP algorithm generates an error of 17 cm after convergence, compared to 2.5 cm when using the particle-free point clouds from the ground truth. The impact of airborne particles on the rotational error is less important than for the translational error with the original point cloud providing comparable performance to the particle-free point clouds. For both translational and rotational errors, removing particles in the LiDAR point cloud using our approaches provides a gain in performance compared to the ground truth.

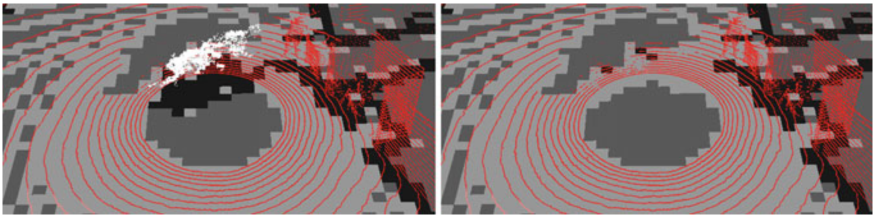


Fig. 8 Occupancy grids (occupied: black cells, free: light grey cells, unobserved: dark grey) and predicted point cloud with *particles* in white and *non-particles* in red for the scene in Fig. 1. Left: Original point cloud. Right: Point cloud with predicted *particles* removed

5.5.2 Obstacle Detection Task

LiDAR sensors are widely used to perform obstacle detection in mobile robotics applications [24]. However, airborne particles in LiDAR point clouds can lead to false obstacles being generated, inducing loss of performance. We evaluate how removing airborne particles in LiDAR point clouds can help obstacle detection in outdoor scenarios using an occupancy grid [4]. The occupancy grid we compute is a 2D array of $30m \times 30m$ and $0.4m$ resolution, with each cell either *unobserved*, *occupied*, or *free*. All cells are initially *unobserved*. A cell is considered *occupied* if a LiDAR point in its 2D boundary is higher than $2m$ or if the height difference between the lowest and highest point is greater than $0.3m$, otherwise it is *free*. Figure 8 shows the difference between an occupancy grid with the original LiDAR points in the presence of airborne particles and an occupancy grid for which the points detected as particles have been removed. Most false obstacles generated by the particles in the original point cloud are removed in the occupancy grid generated from the filtered point cloud. The cells behind the particles are still considered *unobserved*, leaving the same opportunity to the system to later detect an obstacle that might be hidden behind the particles as before the removal of LiDAR points. We evaluate two aspects when filtering LiDAR point clouds for obstacle detection: the percentage of false obstacles removed by correctly identifying the particles, and the percentage of true obstacles removed due to misclassification of a *non-particle* for a *particle*. Table 5 shows the performance of each approach. Both approaches are able to prevent the majority of false obstacles by detecting and removing airborne particles, with the point classification approach preventing up to 96% of false obstacle cells in the occupancy grid. Only a small percentage of true obstacles cells are removed from the occupancy grid, with the voxel classification approach only removing 0.1% of true obstacles.

Table 5 Performance of our classification approaches on the obstacle detection task

	Original point cloud	Voxel classification	Point classification
False obstacle prevented (higher is better)	0%	89%	96%
True obstacle removed (lower is better)	0%	0.1%	0.4%

6 Conclusion and Future Work

In this paper, we proposed two deep-learning approaches to classify airborne particles in LiDAR point clouds, based on voxels and on points (projected into a LIDAR image), respectively. We also presented a thorough validation on realistic experimental dataset containing fog and airborne dust. First, we evaluated the best combination of input features for each approach. The point classification approach performed best using geometry and multi-echo features with an F1 score of 94%, while the voxel classification approach performed best when combining geometry, intensity, and multi-echo features with an F1 score of 91%. Both approaches significantly outperformed the state of the art. These results indicate that the multi-echo returns of the LiDAR sensor are beneficial as an input feature for classifying airborne particles. Both approaches provide comparable classification performance overall; however, the choice between the two strategies depends on the data representation, size of the network. We also show that it may depend on the task to perform. We can recommend the use of the voxel-based approach for a system already using a voxel representation or requiring a smaller network size due to hardware limitation. This approach seems to be well suited for obstacle detection tasks: in this paper, the proposed method was able to clear a significant number of false obstacles (created by airborne particles), while removing only a very small number of true obstacles cells in an occupancy grid. We recommend the point classification approach to get the best possible classification performance at the cost of a bigger network. We also showed that this was the only approach of the two capable of differentiating between types of particles. When filtering a point cloud prior to a pose estimation task, both approaches provide a similar performance gain. Limits of the two approaches are symptomatic of deep learning and given as generalization to other particles, sensors, and environments.

Future work will evaluate the proposed methods on other types of particles such as smoke or snow. Evaluating the uncertainty of the prediction would also be beneficial, as in any robotics application. Finally, adding a notion of time in the network prediction is likely to be useful to learn and predict the movement aspects of airborne particles.

Acknowledgements Parts of the computational resources used in this work were provided by the High Performance Computing Facility of the Queensland University of Technology, Brisbane, Australia. The authors would like to thank Ankit Dhall and Benjamin Le Bigot for their help with data labeling.

References

1. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. CoRR (2016)
2. Darms, M.S., Rybski, P.E., Baker, C., Urmson, C.: Obstacle detection and tracking for the Urban challenge. IEEE Trans. Intell. Transp. Syst. (2009)
3. Dubé, R., Cramariuc, A., Dugas, D., Nieto, J., Siegwart, R., Cadena, C.: SegMap: 3d segment mapping using data-driven descriptors. Robotics: Science and Systems (RSS) (2018)
4. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. IEEE Computer (1989)
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
6. Gerardo-Castro, M.P., Peynot, T., Ramos, F., Fitch, R.: Non-parametric consistency test for multiple-sensing-modality data fusion. In: IEEE International Conference on Information Fusion (2015)
7. Laible, S., Khan, Y.N., Bohlmann, K., Zell, A.: 3D LIDAR- and camera-based terrain classification under different lighting conditions. Auton. Mob. Syst. (2012)
8. Lalonde, J.F., Vandapel, N., Huber, D.F., Hebert, M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility. J. Field Robot. (2006)
9. Levinson, J., Thrun, S.: Robust vehicle localization in urban environments using probabilistic maps. In: IEEE International Conference on Robotics and Automation (2010)
10. Montemerlo, M., Jan, B., Suhrid, B., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D.: Junior: the stanford entry in the Urban challenge. J. Field Robot. (2008)
11. Motaz Khader, S.C.: An introduction to automotive lidar. Tech. rep, Texas Instruments (2018)
12. Peynot, T., Kassir, A.: Laser-camera data discrepancies and reliable perception in outdoor robotics. In: IEEE International Conference on Intelligent Robots and Systems (2010)
13. Peynot, T., Scheding, S., Terho, S.: The marulan data sets: multi-sensor perception in natural environment with challenging conditions. Int. J Robot. Res. (2010)
14. Phillips, T.G., Guenther, N., McAree, P.R.: When the dust settles: the four behaviors of lidar in the presence of fine airborne particulates. J. Field Robot. (2017)
15. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
16. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems (2017)
17. Reymann, C., Lacroix, S.: Improving lidar point cloud classification using intensities and multiple echoes. (2015)
18. Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. Springer Medical Image Computing and Computer-Assisted Intervention (MICCAI) (2015)
19. Rusu, R.B., Cousins, S.: 3D is here: Point cloud library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA) (2011)
20. Shamsudin, A.U., Ohno, K., Westfechtel, T., Takahiro, S., Okada, Y., Tadokoro, S.: Fog removal using laser beam penetration, laser intensity, and geometrical features for 3d measurements in fog-filled room. Adv. Robo.
21. Stanislas, L., Sünderhauf, N., Peynot, T.: Lidar-based detection of airborne particles for robust robot perception. In: Australasian Conference on Robotics and Automation (ACRA) (2018)
22. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. CoRR (2017)
23. Suger, B., Steder, B., Burgard, W.: Terrain-adaptive obstacle detection. IN: IEEE International Conference on Intelligent Robots and Systems (2016)
24. Urmson, C., et al.: Autonomous driving in Urban environments: boss and the Urban challenge. J. Field Robot. (2008)

25. Wurm, K.M., Kümmerle, R., Stachniss, C., Burgard, W.: Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In: IEEE International Conference on Intelligent Robots and Systems (2009)
26. Zhang, C., Luo, W., Urtasun, R.: Efficient convolutions for real-time semantic segmentation of 3d point clouds. In: 2018 International Conference on 3D Vision (3DV) (2018)
27. Zhou, Y., Tuzel, O.: Voxelnet: end-to-end learning for point cloud based 3d object detection. CoRR (2017)

Experimental Validation of Structured Receding Horizon Estimation and Control for Mobile Ground Robot Slip Compensation

Nathan D. Wallace, He Kong, Andrew J. Hill, and Salah Sukkarieh

1 Introduction

Accurate navigation in difficult off-road conditions is one of the major challenges faced in development and deployment of autonomous agents for applications such as farming and environmental monitoring [1]. This is primarily due to the presence of arbitrary unknown disturbances, which are hard to model. In fact, estimation of dynamic systems with unknown inputs has attracted a lot of attention in the systems and control literature. We refer the reader to [2] for a brief overview of the existing literature and more recent progress on this subject. Especially in the context of field robot motion control, the presence of unknown and spatio-temporally varying traction conditions can degrade the performance of traditional control approaches such as PID control [3], which do not account for induced wheel slippage on vehicle handling. It is therefore necessary to estimate the traction parameters online, so that the control algorithm can compensate for wheel slippage and maintain good tracking performance. This is important in situations where the robot is operating near crops or livestock, to ensure sufficient clearance is maintained.

Nonlinear receding horizon estimation and control (RHEC) is one strategy that has been successfully applied to this problem, demonstrated on a variety of platforms, and shown to perform well at speeds up to ∼2 m/s [4–7]. The focus of our prior work is on extending the RHEC framework to improve performance at higher speeds and over more rapidly varying terrains. Building upon insights in [2, 8–10]—and especially the technique in [11]—this is achieved by employing a structured blocking (SB) strategy, which enforces a blocking structure on the estimated slip parameter sequence. The performance of the proposed strategy was demonstrated in simulation to achieve improvements in tracking performance of up to ∼9% [12, 13] when compared to the previous RHEC approaches [4–7], which fix the slip parameters to

---

N. D. Wallace (✉) · H. Kong · A. J. Hill · S. Sukkarieh
Australian Centre for Field Robotics, The University of Sydney, Sydney, NSW 2050, Australia
e-mail: n.wallace@acfr.usyd.edu.au

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_29

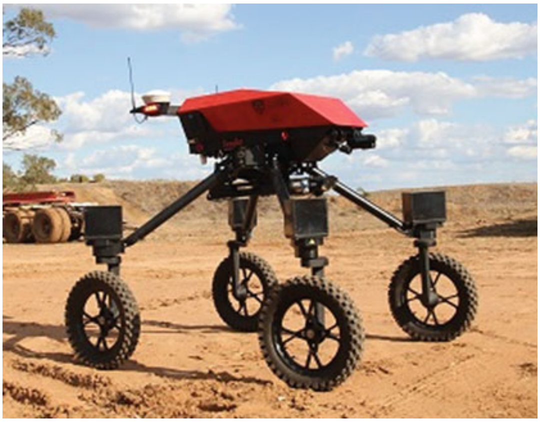


Fig. 1 ACFR's Swagbot platform pictured during a field trial at a cattle station near Nevertire, NSW. Swagbot is a lightweight, electric robot designed to collect data on pasture and livestock, and perform tasks such as autonomous weeding. The platform's rocker chassis design enables it to clamber over uneven terrain and low-lying obstacles

a single value over the estimation horizon—henceforth termed full-horizon blocking (FHB).

With the aim of validating the above method experimentally, in this paper, we implement the nonlinear RHEC approach with structured blocking on the Australian Centre for Field Robotics' (ACFR) Swagbot electric ground vehicle—shown in Fig. 1—for trajectory tracking in unknown and variable slip environments.

We compare the results of our method against the FHB strategy adopted in [4–7], and show that the RHEC with SB achieves improved path tracking performance compared to the FHB method, with a reduction of up to 7% in RMSE. Speed tracking at 2.5 m/s was significantly improved also, with a 13% reduction in RMSE, although a marginal reduction in speed tracking performance was observed for the 1.5 m/s test. Better tracking performance is important for time and energy efficient operation of the mobile platform, in conjunction with data-driven motion cost models [14]. Further tests are planned to better characterise and validate control performance in more challenging environments.

2 Related Work

There is an extensive body of literature on slip estimation for wheeled mobile robots (WMRs), covering approaches such as bounded uncertainty models, extended kinematic models (KMs) which account for slip effects, machine learning models and dynamic models. We focus primarily on the extended KM approach here, as it is the most relevant to the presented work. A more thorough review of the other methods can be found in our previous works [12, 13].

Extended Kalman filters (EKFs) with kinematic constraints have been used for slip estimation in WMRs, with slip modelled as a white-noise velocity perturbation [15] or as a multiplicative factor within the KM for a skid-steered WMR [16]. In

[17], a tractor-trailer system with steering slip is modelled by generalised bicycle kinematics with an additional multiplicative factor, using an EKF to estimate state and slip parameters, and a nonlinear RHC for path tracking. Lateral tracking errors of <15 cm on straight and gently curved paths are achieved at 3.3 m/s.

A similar method for modelling slip is adopted in [4] for a tractor, augmenting a bicycle KM with two multiplicative factors to capture longitudinal and side slip (rather, slip in the steering angle)—which are assumed to be constant over the estimation horizon. Due to the existence of state and parameter constraints, and a nonlinear measurement function, a more sophisticated receding horizon estimation (RHE) strategy was required. A receding horizon control (RHC) strategy was used to control the wheel velocity and steering rate, and this RHEC framework is tested experimentally on wet, uneven grass, with average tracking errors of 0.26 m on gentle curves, and 0.6 m in headland turns at ~2 m/s.

Similar approaches have been applied to tracked and skid-steered WMRs in [5, 6], using a unicycle KM with factors for longitudinal and lateral slip, achieving tracking errors of <12 cm at ~0.2 m/s for the tracked, and <5 cm at ~0.3 m/s for the skid-steered robot. A tractor-trailer system was modelled in [7] by a generalised tricycle KM with three factors for longitudinal and slide slip. Centralised RHE was used, and the distributed RHC framework is compared against a centralised strategy, achieving comparable tracking errors of <10 cm with faster solution times.

Recently, the above RHEC framework was compared against a linear RHC approach in [18], using input-state linearisation with an EKF estimator to compensate for system state nonlinearity. It is shown that RHEC produces improved trajectory tracking performance over the input-state linearisation approach, although at a slight increase in computation cost.

A tractor is modelled in [19] by an extended bicycle KM with additive side-slip effects on both wheels—ignoring longitudinal slip. A nonlinear adaptive control law embedded with an RHC algorithm was designed to avoid tracking overshoot, achieving <15 cm tracking errors with velocity manually controlled, though subsequent observations in [20] revealed decreased performance as vehicle speed increases. A mixed backstepping kinematic and dynamic observer was developed to improve slip observation, improving tracking performance at speeds of up to 5 m/s.

3 Preliminaries

Notation: Throughout this paper, $[a_1, \ldots, a_n]$ denotes $\left[a_1^T, \ldots, a_n^T\right]^T$, where $a_1, \ldots,$ a_n are appropriately dimensioned scalars/vectors/matrices. The weighted Euclidean norm is denoted by $\|a\|_R^2 = a^T Ra$. Subscripts i, j, k denote the value of that variable at time t_k, i.e. $a_k = a(t_k)$. $\mathscr{I}_i^j$ denotes the set of integers between i and j.

3.1 System Model

The vehicle model used in this work is an adaptation of the bicycle KM, which is known to approximate car-like vehicle behaviour quite accurately [21–23]. Slip is incorporated in the model via two multiplicative slip parameters; one for longitudinal slip, and another for steering slip.

This steering slip formulation was adopted in [17] to capture the effect of vehicle inertia on the steering angle, yielding a slipping angle proportional to the steering angle. This allows the slip parameter to remain somewhat constant regardless of the steering motion, whereas an additive slip parameter would change value constantly—especially in curves. This model assumes that there are no external forces acting in the sideways direction on the vehicle, which is typically the case, save for sharp turns at high speeds and steep inclines.

The equations describing the robot's motion are as follows:

$$\begin{bmatrix} \dot{x}_{pos} \\ \dot{y}_{pos} \\ \dot{\beta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \kappa u_1 \cos \beta \\ \kappa u_1 \sin \beta \\ \frac{\kappa u_1}{l} \tan (\mu \cdot \delta) \\ u_2 \end{bmatrix}, \tag{1}$$

where x_{pos} [m] and y_{pos} [m] are the position of the virtual 'rear wheel'—or centre of the rear axle in the case of a four-wheeled vehicle—in global Cartesian coordinates, β [rad] is the yaw angle or heading, and δ [rad] is the steering angle; shown in Fig. 2. The control variables are the wheel speed with respect to the ground u_1 [m/s], and the steering rate u_2 [rad/s]. The parameter l [m] represents the wheel base.

The parameters κ, μ represent the longitudinal and steering slip, respectively. κ relates commanded wheel speed to actual ground speed, capturing the influence of slip, and any error or uncertainty in the vehicle odometry. μ results from

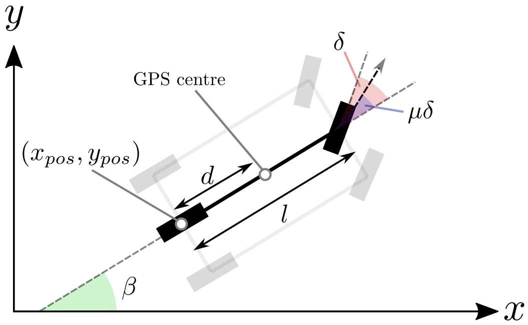


Fig. 2 Schematic illustration of adapted bicycle model with steering slip affecting the front wheel. The steering angle δ is shown in red, the effective steering angle $\mu\delta$ is shown in blue and the vehicle heading β is shown in green. The GPS centre is offset by a distance d from the rear-axle centre on the Swagbot platform

vehicle inertia's impact on steering, which is approximately relative to the steering angle. The parameter value range is conservatively chosen to be [0, 1]—assuming no over-steering and no negative slip. Though circumstances do exist under which this assumption does not hold, for the environment and speeds tested, this assumption closely matched observations, and enforcing these bounds ensured the problem solution was both well-behaved, and quickly solvable. The choice of domain also allows the percentage slip experienced to be expressed as $1 - \kappa$ and $1 - \mu$.

It is our goal to eventually accommodate the effects of lateral slip in our model, either by introducing a side-slip term or adopting a more sophisticated four-wheel independent steer/drive model. Both approaches increase the difficulty of the optimisation problem, and in adding additional unobservable parameters, could risk compromising the estimates of each. That said, for the conditions and speeds featured in our experiments, the robot's momentum was insufficient for significant lateral slippage to occur, and thus the primary slip contributions were due to longitudinal and steering slip. For this reason, and for ease of comparison with the existing motion models used in related literature, we chose to ignore the lateral slip component.

4 Method

4.1 Receding Horizon Estimation and Control

An RHEC strategy is adopted in this work, which consists of solving two least-squares optimisation problems—one for estimation, and one for predictive control—resulting in a closed-loop control strategy.

It should be noted that while we detail the standard RHC case in this section, a *packeted RHC* approach was implemented on the platform, which publishes the entire control horizon, rather than just the next action. This improves robustness to time overrun by ensuring suitable actions are available even when a control iteration is missed. The packeted approach is otherwise identical to standard RHC. In our testing, time overrun was only observed during the first 1–2 s of operation; a consequence of the initialisation overhead of internal MATLAB functions.

The nonlinear RHEC problem is formulated as follows. Let $x_k \in \mathbb{R}^{n_x}$ represent the state of the system, and $y_k \in \mathbb{R}^{n_y}$ be an observation of this system state—here, the cartesian position and heading of the robot in world coordinates, and the steering angle. Let $u_k \in \mathbb{R}^{n_u}$ represent the control actions taken—the commanded forward speed and steering angle—and $p_k \in \mathbb{R}^{n_p}$ the system parameters to be estimated, which in our case are the longitudinal and steering slip experienced by the robot. We assume the system evolves in accordance to the given dynamic model $\dot{x}_k = f(x_k, u_k, p_k)$, and observations are taken in accordance with the measurement model function $y_k = h(x_k, u_k, p_k)$. Let variables (x_k, y_k, u_k, p_k) refer to the *real* process. These each have associated *decision* and *optimal decision* variables in the optimisation, which we denote $(\chi_k, \eta_k, \nu_k, \rho_k)$ and $\left(\hat{x}_k, \hat{y}_k, \hat{u}_k, \hat{p}_k\right)$, respectively. The values of these decision variables are iteratively varied, subject to the constraints, by the optimiser, eventually yielding the optimal decision once the objective is minimised.

At each sampling time t_k, let the real system state be x_k. A sensor measurement y_k is taken, and the RHE uses the past N_e measurements $y_j,\ j \in \mathscr{I}_{k-N_e+1}^{k}$ to estimate (x_k, p_k). This estimate $\left(\hat{x}_k,\ \hat{p}_k\right)$ is used by the RHC module along with a time-based reference trajectory $\lambda_j = \left[\lambda_j^x, \lambda_j^u\right],\ \lambda_j^x \subseteq \mathbb{R}^{n_x}, \lambda_k^u \subseteq \mathbb{R}^{n_u}, j \in \mathscr{I}_{k+1}^{k+N_c}$. The RHC module computes the optimal control actions over a finite forward horizon of N_c timesteps, and the next action $\hat{u}_k$ is executed by the robot. This process repeats each sampling time (Δt seconds apart) until the end of the trajectory is reached.

4.2 Nonlinear Receding Horizon Estimation

RHE is a powerful optimisation-based estimation technique that provides a systematic framework for handling constraints and nonlinearities. In contrast to full information estimation (FIE)—which utilises the entire history of available information—RHE uses only measurements taken within a finite time-frame, and captures the information of prior measurements in an arrival cost term.

Given N_e past measurements $y_j,\ j \in \mathscr{I}_{k-N_e+1}^{k}$ and a measurement model function h, the constrained estimation problem to be solved at time t_k is

$$\min_{\chi(\cdot),\nu(\cdot),\rho(\cdot)} \quad \left\{ \Gamma_{k-N_e+1} + \sum_{i=k-N_e+1}^{k} \left\| \eta_i - y_i \right\|_{R_k}^2 \right\} \tag{2}$$

$$s.t. \quad \left.\begin{array}{l} \dot{\chi}_j = f\left(\chi_j, \nu_j, \rho_j\right) \\ \eta_j = h\left(\chi_j, \nu_j, \rho_j\right) \\ \chi_{min} \leq \chi_j \leq \chi_{max} \\ \rho_{min,j} \leq \rho_j \leq \rho_{max,j} \end{array}\right\} j \in \mathscr{I}_{k-N_e+1}^{k}$$

$$\Gamma_{k-N_e+1} = \left\| \begin{array}{l} \chi_{k-N_e+1} - \hat{x}_{k-N_e+1} \\ \rho_{k-N_e+1} - \hat{p}_{k-N_e+1} \end{array} \right\|_{\Pi_{k-N_e+1}^{-1}}^2 \tag{3}$$

where Γ_{k-N_e+1} is the arrival cost function, $\hat{x}_{k-N_e+1} = \hat{x}_{k-N_e+1|k-N_e}$ is the optimal state prediction at t_{k-N_e+1}, R_k is the symmetric positive semi-definite weighting matrix equal to the inverse of the measurement noise covariance matrix [24], and Π^{-1} is the inverted covariance matrix of a smoothed EKF—as considered in [4, 25].

In the context of localisation and slip estimation, this RHE problem chooses the sequence of positions, control actions and slip values which minimise the least-squares distance between the extended KM and the measured robot state over the horizon. Consideration of the motion constraints allows the optimiser to estimate the extent of the slippage the robot's motion was subject to.

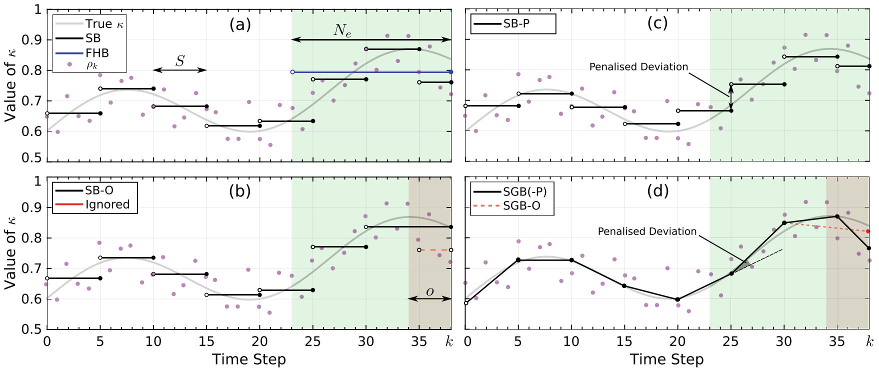


Fig. 3 Illustration of the SB extensions. The true κ value—not typically observable, shown here purely for illustrative purposes—is shown in light grey, and the best fit parameter estimate ρ_k based on noisy measurement data is shown in purple. **a** The FIE SB structure for block size $S = 5$; the RHE observes only the last N_e time steps. The equivalent FHB block is shown in blue. **b** SB with an overlap $o = 4$. The terminal block in the sequence (red) has size $3 < o$ and thus is suppressed. **c** SB with block value deviations penalised. **d** SGB variants

4.3 Structured Noise Blocking in Estimation

One of the main limitations inherent in constrained estimation methods such as RHE—and especially FIE—is that an optimisation problem must be solved at each sampling time. This burden can be reduced either via reformulation as a multi-parametric quadratic programme [26], exploiting the optimisation structure [27], or via the structured blocking strategy introduced in our previous work [11–13].

The SB strategy has recently been adopted in FIE and RHE for estimating the process noise sequence [11], and more importantly, it was shown that the segment structure of the noise sequence from SB FIE must also be enforced over the finite estimation horizon of RHE (i.e. steps between $T - N$ and $T - 1$). The resulting SB RHE strategy becomes a dynamic estimator with periodically varying computational complexity. An illustrative example of this concept is shown in Fig. 3a.

Our recent work applies this strategy to nonlinear RHEC. In [13], it was observed that while SB improved state estimation quality, the parameter estimates in the terminal block—used for the following RHC step—fluctuated in quality, due to the changing size of the terminal block in the receding horizon. A SB with overlapping blocks (SB-O) strategy was proposed to ameliorate this issue, enabling a trade-off between state and parameter estimate quality via an overlap extent parameter o. An example of this strategy is shown in Fig. 3b.

A number of complementary strategies for SB RHEC were also introduced in [12], including the addition of a value-shift penalty between consecutive blocks (SB-P)—shown in Fig. 3c—and a strategy combining this with SB-O, termed SB-OP. Furthermore, gradient blocking methods were also proposed, which, in contrast to the aforementioned value-blocking methods, instead perform blocking on the first-derivative of the parameter sequence. An example is shown in Fig. 3d.

4.4 Nonlinear Receding Horizon Control

The RHC algorithm seeks to predict the system behaviour over a finite time horizon via minimisation of a cost function composed of states, inputs and references, and similarly to RHE, the RHC framework also supports handling of state and input constraints.

For RHC, at the current time t_k, we wish to predict the control sequence for the next N_c time steps; thus the constrained optimisation problem to be solved is

$$\begin{aligned} \min_{\nu(\cdot)} \quad & \left\{ \sum_{i=k+1}^{k+N_c-1} \left\| \begin{matrix} \lambda_i^x - \chi_i^\lambda \\ \lambda_i^u - \nu_i^\lambda \end{matrix} \right\|_{V_k}^2 + \Omega_{k+N_c} \right\} \\ s.t. \quad & \left. \begin{matrix} \dot{\chi}_j = f\left(\chi_j, \nu_j, \rho_j\right) \\ \chi_{min} \leq \chi_j \leq \chi_{max} \\ \nu_{min} \leq \nu_j \leq \nu_{max} \end{matrix} \right\} \; j \in \mathcal{I}_{k+1}^{k+N_c} \end{aligned} \tag{4}$$

$$\Omega_{k+N_c} = \left\| \lambda_{k+N_c} - \chi_{k+N_c}^\lambda \right\|_{V_N}^2 \tag{5}$$

where Ω_{k+N_c} is the terminal cost function, $\chi_k^\lambda \in \mathbb{R}^{n_\lambda}$ is the subset of the state variables χ_k with corresponding entries in λ_k, and V_k, V_N are the symmetric positive-semi-definite weighting matrices.

In the context of path tracking, the RHC problem chooses the sequence of control actions which, given the current position and slip parameter values, minimises the least-squares distance between the model predicted motion and the target reference path over the horizon.

4.5 Solution Methods

Both RHE (2) and RHC (4) are nonlinear least-squares optimisation problems, so similar methods can be used to solve both. In this work, the generalised Gauss–Newton multiple-shooting method—derived from the classical Newton method—is used. This method was developed specifically for solving least-squares optimisation problems quickly, by utilising the real-time iteration scheme proposed in [28], which improves convergence and reduces computation cost.

To solve the constrained nonlinear optimisation problems involved in RHEC, the *ACADO* code-generation tool is used, which is an open-source software package which generates customised real-time RHE and RHC algorithms as efficient, self-contained C-code modules [29]. The generated dense quadratic sub-problems are then solved using the *qpOASES* online quadratic programme solver [30].

5 Experimental Setup

The objective in these real-time experiments is to demonstrate the performance of the proposed RHEC with SB strategy for accurately following a user-defined trajectory (waypoints and speed) over a grass field on a slight incline—an example of the scenario described can be seen in Fig. 4. We note that while the chosen test site is homogeneous in terms of terrain type, it does contain a steady incline, which we expect to impact slip in a more systematic and controlled manner as opposed to changing the type of terrain—an important property when attempting to characterise performance during preliminary testing.

The platform used in these experiments is the ACFR Swagbot (Fig. 1). The measurements available from this platform include the GPS position and orientation, the angles of the steering joints, and the angular rate of the drive joints. The joint angles are used to calculate the virtual centre steering angle, steering rate and speed. The pose is provided by the Trimble BD982 GNSS, and the joint measurements by encoders and Hall-effect sensors for the steering and drive joints, respectively.

The system runs using a combination of MATLAB, Python and ACFR internal software modules, with inter-communication handled using ROS. The RHEC algorithm is implemented in MATLAB, receiving sensor readings via custom ROS buffer messages, and publishing the sequence of control commands over the forward horizon to an Ackermann module, which emulates Ackermann steering behaviour on the

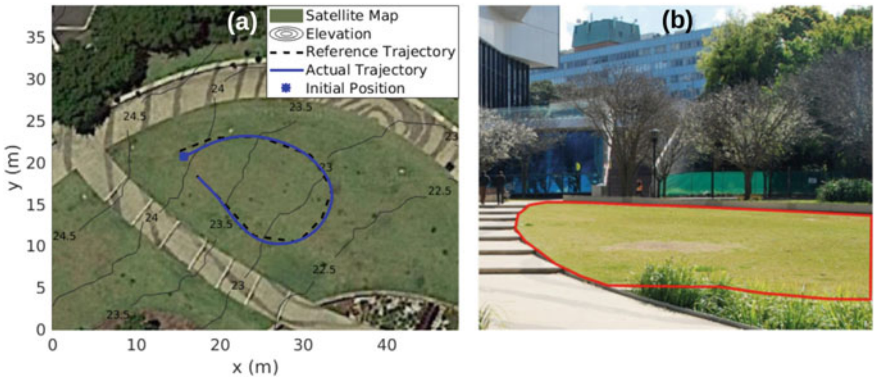


Fig. 4 **a** Reference (black) and actual (blue) trajectory followed by Swagbot using the SB RHEC strategy. Contour map displays the elevation in metres. **b** A photograph of the testing area—the region used for the experiments is highlighted with a red border

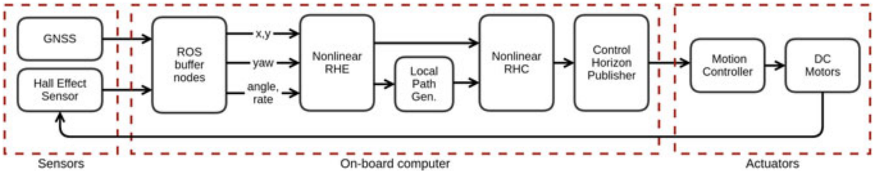


Fig. 5 System diagram of the RHEC implementation on the Swagbot platform

Swagbot platform. The system diagram describing the sensing-estimation-control-actuation pipeline as implemented on the robot is shown in Fig. 5.

The GPS centre is offset to correspond to the robot body frame centre, whereas the bicycle model position is defined to be the centre of the rear axle—which lies at a distance of $d = 0.64$ m directly behind the robot centre. Thus the position measurements at the rear axle vary nonlinearly with the platform heading β.

Conservative estimates of variance for the sensor measurements were used $\sigma_x = \sigma_y = 0.12$ m for the GPS position measurement and $\sigma_\beta = 0.035$ for the heading (based on the GPS accuracy on the day), $\sigma_\delta = 0.035$ for the steering angle, and $\sigma_{u_1} = 0.1$ m/s and $\sigma_{u_2} = 0.17$ rad/s for the actuation measurements.

The RHE and RHC modules are implemented in ACADO and MATLAB. For simplicity, we opt to use similar weightings as employed in [4]. The horizon length and step size used are $N_e = N_c = 15$ and $\Delta t = 0.2$ s, respectively, to ensure sufficient spatial distance between consecutive sampling points, and a long enough horizon to allow reliable convergence to the reference path. The steering angle is constrained to $-45° \leq \delta \leq 45°$, and the initial guesses for the estimated slip parameter values are $\kappa_0 = \mu_0 = 0.5$.

5.1 RHE Configuration

The measurement function used in these experiments is $h\,(x, u) = [x_{pos} + d\cos(\beta), y_{pos} + d\sin(\beta), \beta, \delta, u_1, u_2]^T$, with d being the offset defined in Sect. 5. The measurement variances are used in the calculation of the RHE weighting matrix, $R_k = \text{diag}\left(\sigma_x^2, \sigma_y^2, \sigma_\beta^2, \sigma_\delta^2, \sigma_{u_1}^2, \sigma_{u_2}^2\right)^{-1} \in \mathbb{R}^{6\times 6}$.

To enforce the desired blocking structure within the ACADO solver—as ACADO does not yet explicitly support parameter estimation—the state and system model (1) is augmented with the slip parameter terms and two corresponding virtual controls—u_κ, u_μ—as follows: $\left[\ldots, \dot{\kappa}, \dot{\mu}\right]^T = \left[\ldots, u_\kappa, u_\mu\right]^T$. The virtual controls are introduced to control the rate of change of the slip parameters κ and μ, which are constrained to be zero everywhere except at the start of a blocking interval.

In the FHB case, u_κ, u_μ are set to zero, fixing the parameter value over the horizon. To implement block overlapping, two blocking parameters S and o are introduced, specifying block size and overlap extent, respectively. At each sampling time, the bounds for u_κ, u_μ are set to be zero everywhere except at the start of a new block, both defining and enforcing the blocking structure. The overlap constraint is enforced by suppressing the formation of new blocks after time $t_k - o\Delta t$.

To penalise the parameter value shift between blocks, the virtual control terms u_κ, u_μ are added to the measurement function, $h(x, u) = \left[\cdots, u_\kappa, u_\mu\right]^T$, and corresponding weighting terms w_{u_κ}, w_{u_μ} are added to R_k. The value of these weighting terms dictates the extent to which block value deviations are penalised relative to the other measured values. The value of the virtual u_κ, u_μ measurements are always set to zero, thus penalising the magnitude of the value shift. Additionally, the shift in parameter values between blocks is constrained to be $-1/\Delta t \leq \dot{\kappa}, \dot{\mu} \leq 1/\Delta t$—the maximum shift in value possible within the bounds.

For the experiments conducted, the SB-OP RHEC variant is used, with a block-size and overlap $S = o = 3$, and penalty weightings of $w_{u_\kappa} = w_{u_\mu} = 10$. We have considered only the value-blocking variant in this paper, due to its more consistent performance across all speeds.

It should be noted that persistent excitation is required to accurately estimate the traction parameters, since the mechanism by which these parameters are observed is proprioceptive in nature. As the robot's speed and steering angle approach zero, these parameters will become unobservable.

5.2 RHC Configuration

A 'global' space-based reference trajectory $\Lambda_j = [x_{ref}^{(j)}, y_{ref}^{(j)}, u_{1,ref}^{(j)}, u_{2,ref}^{(j)}]^T$ consists of a sequence of waypoints augmented with a desired control reference. At each t_k, a new time-based 'local' trajectory λ is generated from Λ, as defined in Sect. 4.1.

This is achieved by taking the current position estimate and finding the closest point $(\tilde{x}_{ref}, \tilde{y}_{ref})$ on the global trajectory. The next N_c points along the trajectory, spaced equally $\Delta t \cdot \tilde{u}_{ref}^k$ metres along the path, are then returned as the local trajectory λ_k. Interpolation is performed to obtain the associated control values $(\tilde{u}_{1,ref}, \tilde{u}_{2,ref})$. The weighting matrices for the RHC step are $V_k = \text{diag}(1.0, 1.0, 5.0, 5.0)$ and $V_T = 10 \cdot V_k$, and the steering rate constraints are $-45°/s \leq u_2 \leq 45°/s$.

6 Results and Discussion

Despite being initialised in a slightly off-track position, Swagbot was observed to quickly converge to the reference trajectory and remain on-track thereafter. For comparison of the reference and actual trajectories of the robot, see Fig. 4a. This trajectory was tracked one time each for the FHB and SB-OP methods at 1.5 m/s, and again at 2.5 m/s. The trajectory was manually specified by selecting waypoints, and as such is composed of straight line segments with abupt turns, which are impossible to track

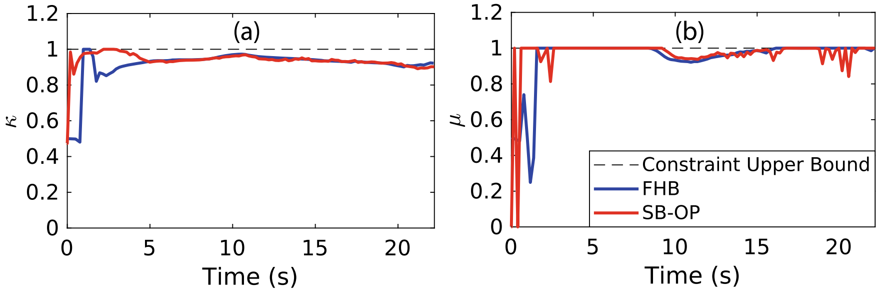


Fig. 6 Estimated slip parameter values for RHE at 2.5 m/s. **a** Estimated κ value. **b** Estimated μ value. Black dashed line denotes parameter upper bound of 1

perfectly with an Ackermann vehicle. The controller instead produces a smoother trajectory, managing to remain close to the reference while also minimising steering effort. It is worth noting that while the infeasible reference path will result in a larger tracking error compared to an equivalent clothoid path, it does present a more challenging tracking problem for the controller.

The packeted RHC method allows the system to continue tracking even if a sampling time deadline is missed, though this behaviour was only observed in the first 1–2 s of the tracking, due to initialisation overhead. As a safeguard, if more than a second passes without an update, or if the estimator or controller fails, the robot is immediately commanded to stop.

An example of the slip parameter estimates for the 2.5 m/s runs can be seen in Fig. 6. We observe that the SB-OP strategy is able to recover far more quickly from the (intentionally) poor initial slip estimate than the FHB strategy, rapidly converging to the expected slip values, albeit with some initial oscillation in μ. Overall, the values for both parameters tended to fall within the 0.8–1 range. While slip bounds of 0–1 were chosen as the parameter constraints, these were fairly conservative for the conditions, and these bounds may be relaxed or tightened to suit the environment. Tighter ranges would serve to increase speed of convergence, and reduce risk of instability due to noise and outlier measurements, though the model would then have difficulty accounting for motions resulting from slippage outside the described range.

The value of κ gradually decreased over the course of the run, indicating an increase in slip correllating with the gradual increase in slope along the trajectory (from roughly $-5°$ to $5°$). The large initial oscillations in the μ estimate were expected, as the steering slip is unobservable at the initial steering angle and rate of zero. Once turn commands are given, the value quickly converges to a more reasonable estimate. We do see an increase in steering slip occurring at the 9–15 s interval, which corresponds to when the sharpest turn is negotiated.

In terms of path tracking, the impact of SB-OP's improved slip estimation performance can best be seen in the 0–5 s interval of Fig. 7a, where the strategy recovers quickly from a larger initial position error, and displays better position tracking performance up until the point that the two strategies' estimates converge. This clearly

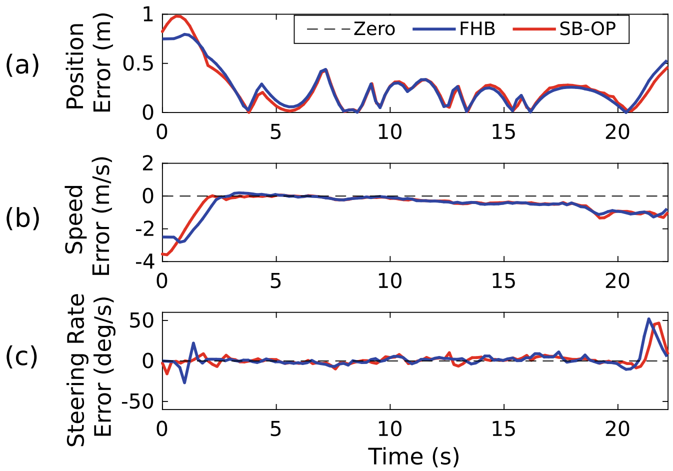


Fig. 7 Tracking error comparison between FHB and SB-OP at 2.5 m/s. **a** Position error, measured as distance from closest point on reference path. **b** Speed (u_1) error. **c** Steering rate (u_2) error. Note that the purpose of the steering rate tracking is to minimise steering effort, rather than perfectly track a given profile (i.e. $u_{2,ref} = 0$), thus zero error only occurs when not actively turning

demonstrates SB-OP's improved ability to compensate for sudden and significant changes in slip conditions. This same trend is also reflected in both the speed tracking and steering effort minimisation, as shown in Fig. 7b and c, respectively.

There is no significant drift in the magnitude of the position error over the run (Fig. 7a), as would be expected in the presence of nonuniform slip conditions, indicating that the controller has successfully compensated for the increased slippage in the uphill section; this is also mirrored by the decrease in estimated κ in Fig. 6a. We do, however, observe an increase in speed tracking error (Fig. 7b) for this segment, indicating that the longitudinal slip experienced is not wholly being compensated for. We suspect the reason for this lies in our overly conservative initial estimate for the wheel speed measurement covariance, which lowers the weighting of this term in the optimisation problem, and thus limits how aggressively the speed error is minimised. Further characterisation of the speed sensing pipeline is planned in order to provide a more realistic covariance estimate.

The tracking performance of the methods at each tested speed is summarised in Fig. 8, displaying RMSE for the position and speed; adjusting for the discrepancy in starting displacement. The improved slip estimation afforded by the SB-OP strategy is reflected by a 4–7% reduction in root-mean-squared error (RMSE); an overall improvement in tracking performance comparable with our simulation results.

In terms of speed tracking, a significant improvement was observed at 2.5 m/s, with $\sim$13% reduction in RMSE. For 1.5 m/s, the speed tracking of SB-OP was slightly worse than FHB, and we suspect that this may be due to the increased susceptability to measurement noise at these lower speeds, where the slip variation over the horizon is less significant, and thus better approximated by a larger block size.

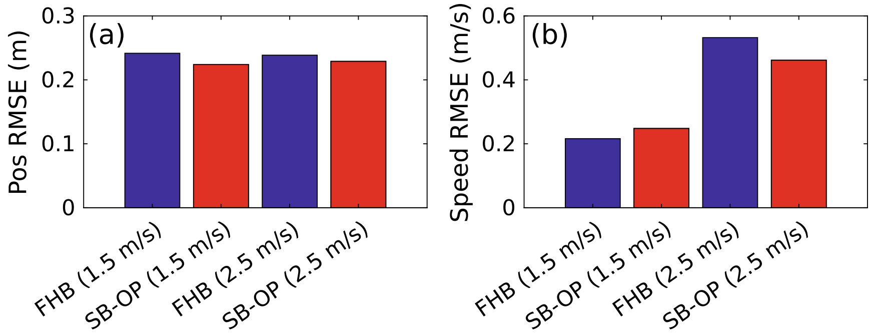


Fig. 8 Tracking RMSE for each run, for **a** the position, and **b** the speed

Optimal selection of N_e, N_c, S, o, w_{u_κ}, w_{u_μ} is guaranteed to yield tracking performance at least as good as FHB in all cases, as FHB is a special case of SB-OP (where $o = N_e$)—the reduced performance seen here is a result of a suboptimal manual selection of these parameters. At lower speeds, the spatial interval between successive samples—and consequently, the spatial extent of the horizon—is reduced substantially, and thus measurement noise becomes a larger contributor to variance between the samples, compromising parameter estimation accuracy. One way to resolve this issue involves increasing the sampling interval, and thereby the spatial span of the horizon, however, sparse sampling is undesirable in a control context, as it limits the controller's ability to adapt quickly to changes in conditions. With SB, it is possible to increase the horizon length without adjusting the sampling interval, whilst modulating the block size S to curtail the number of parameters to estimate, and thus the complexity of the problem, emulating the benefits of larger sampling intervals without sacrificing controllability.

Similarly, o encodes the minimum spatial extent necessary for a good current slip estimate, and w_{u_κ}, w_{u_μ} encode the expected rate of spatial variance of surface slip in the environment. All these quantities can reasonably be selected as a function of platform speed, requiring only determination of a sufficient spatial interval for accurate parameter estimation. Thus only one design parameter would be required to determine a near optimal configuration for the SB-OP across the full range of operating speeds. Optimal selection of o, w_{u_κ}, w_{u_μ}, however, would also be dependent on the roughness (spatial slip variance) of the terrain.

7 Conclusion

An RHEC algorithm for tracking trajectories in slippery environments was implemented on the Swagbot platform and preliminary testing of the strategy was performed. Both FHB and SB methods were tested, and their performance were compared. Results demonstrated SB improved tracking performance by 4–7% over FHB.

Future work will involve further testing and validation on more varied and challenging terrain conditions, optimal blocking parameter and horizon selection as a function of speed, and application of the RHEC SB strategy with a more complex four-wheel independent steer/drive model.

References

1. Li, M., Imou, K., Wakabayashi, K., Yokoyama, S.: Review of research on agricultural vehicle autonomous guidance. Int. J. Agric. Biol. Eng. **2**(3), 1–16 (2009)
2. Kong, H., Sukkarieh, S.: An internal model approach to estimation of systems with arbitrary unknown inputs. Automatica (2019) (accepted and to appear)
3. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive tracking control of a nonholonomic mobile robot. IEEE Trans. Robot. Autom. **16**(5), 609–615 (2000)
4. Kraus, T., Ferreau, H.J., Kayacan, E., Ramon, H., De Baerdemaeker, J., Diehl, M., Saeys, W.: Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. Comput. Electron. Agric. **98**, 25–33 (2013)
5. Kayacan, E., Young, S.N., Peschel, J.M., Chowdhary, G.: High-precision control of tracked field robots in the presence of unknown traction coefficients. J. Field Robot. 1–13 (2018)
6. Kayacan, E., Zhang, Z., Chowdhary, G.: Embedded high precision control and corn stand counting algorithms for an ultra-compact 3D printed field robot. In: Proceedings of Robotics: Science and Systems, pp. 1–8, Pittsburgh, Pennsylvania (2018)
7. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Distributed nonlinear model predictive control of an autonomous tractor-trailer system. Mechatronics **24**(8), 926–933 (2014)
8. Kong, H., Goodwin, G., Seron, M.M.: Predictive metamorphic control. Automatica **49**, 3670–3676 (2013)
9. Goodwin, G.C., Kong, H., Mirzaeva, G., Seron, M.: Robust model predictive control: reflections and opportunities. J. Control Decis. **1**(2), 115–148 (2014)
10. Kong, H., Sukkarieh, S.: Metamorphic moving horizon estimation. Automatica **97**, 167–171 (2018)
11. Kong, H., Sukkarieh, S.: Suboptimal receding horizon estimation via noise blocking. Automatica **98**, 66–75 (2018)
12. Wallace, N.D., Kong, H., Hill, A.J., Sukkarieh, S.: Structured noise blocking strategies for receding horizon estimation and control of mobile robots with slip. In: Proceedings of Australasian Conference on Robotics and Automation, pp. 1–10, Lincoln, New Zealand (2018)
13. Wallace, N.D., Kong, H., Hill, A.J., Sukkarieh, S.: Receding horizon estimation and control with structured noise blocking for mobile robot slip compensation. In: 2019 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–7, Montreal, Canada (2019)
14. Wallace, N.D., Kong, H., Hill, A.J., Sukkarieh, S.: Motion cost characterisation of an omnidirectional WMR on uneven terrains. In: Proceedings of Joint 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles and 1st IFAC workshop on Robot Control, Daejeon, Korea (2019) (accepted and to appear)
15. Dissanayake, G., Sukkarieh, S., Nebot, E., Durrant-Whyte, H.: The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. IEEE Trans. Robot. Autom. **17**(5), 731–747 (2001)
16. Yi, J., Wang, H., Zhang, J., Song, D., Jayasuriya, S., Liu, J.: Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. IEEE Trans. Robot. **25**(5), 1087–1097 (2009)
17. Backman, J., Oksanen, T., Visala, A.: Navigation system for agricultural machines: nonlinear model predictive path tracking. Comput. Electron. Agric. **82**, 32–43 (2012)

18. Kayacan, E., Saeys, W., Ramon, H., Belta, C., Peschel, J.M.: Experimental validation of linear and nonlinear MPC on an articulated unmanned ground vehicle. IEEE/ASME Trans. Mechatron. **23**(5), 2023–2030 (2018)
19. Lenain, R., Thuilot, B., Cariou, C., Martinet, P.: High accuracy path tracking for vehicles in presence of sliding: application to farm vehicle automatic guidance for agricultural tasks. Auton. Robots **21**(1), 79–97 (2006)
20. Lenain, R., Thuilot, B., Cariou, C., Martinet, P.: Mixed kinematic and dynamic sideslip angle observer for accurate control of fast off-road mobile robots. J. Field Robot. **27**(2), 181–196 (2010)
21. LaValle, S.M.: Planning Algorithms, pp. 722–726. Cambridge University Press (2006)
22. Snider, J.M.: Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08 (2009)
23. Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F.: Kinematic and dynamic vehicle models for autonomous driving control design. In: 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 1094–1099, Seoul, South Korea (2015)
24. Ferreau, H., Kraus, T., Vukov, M., Saeys, W., Diehl, M.: High-speed moving horizon estimation based on automatic code generation. In: 2012 IEEE 51st Annual Conference on Decision and Control (CDC), pp. 687–692, Maui, HI (2012)
25. Rao, C.V., Rawlings, J.B., Lee, J.H.: Constrained linear state estimation-a moving horizon approach. Automatica **37**(10), 1619–1628 (2001)
26. Voelker, A., Kouramas, K., Pistikopoulos, E.: Moving horizon estimation: error dynamics and bounding error sets for robust control. Automatica **49**(4), 943–948 (2013)
27. Morabito, B., Kogel, M., Bullinger, E., Pannocchia, G., Findeisen, R.: Simple and efficient moving horizon estimation based on the fast gradient method. In: Proceedings of 5th IFAC Conference on NMPC, pp. 428–433, Seville, Spain (2015)
28. Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., Allgöwer, F.: Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. J. Process Control **12**(4), 577–585 (2002)
29. Houska, B., Ferreau, H.J., Diehl, M.: ACADO toolkit—an open source framework for automatic control and dynamic optimization. Optim. Control Appl. Methods **32**(3), 298–312 (2011)
30. Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., Diehl, M.: qpOASES: a parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014)

Lessons Learned from Deploying Autonomous Vehicles at UC San Diego

David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit Binnani, and Henrik I. Christensen

1 Introduction

The Autonomous Vehicle Laboratory (AVL) at UC San Diego's Contextual Robotics Institute has partnered with UC San Diego's Logistics Unit, Police Department, and Mail Delivery Center to develop and deploy self-driving carts on campus. At the moment, two GEM e6 development platforms are being used to perform tests around the campus. During the initial development stage, the vehicles are expected to perform mail and package deliveries at different campus locations in a variety of stochastic traffic and road conditions. In the second development stage, the team will lead the effort on last-mile transportation for campus visitors, students, and faculty

---

D. Paz (✉) · P.-J. Lai · S. Harish · H. Zhang · N. Chan · C. Hu · S. Binnani · H. I. Christensen
UC San Diego - 9500 Gilman Dr, La Jolla, CA 92093, USA
e-mail: dpazruiz@ucsd.edu

P.-J. Lai
e-mail: polai@ucsd.edu

S. Harish
e-mail: ssumuka@ucsd.edu

H. Zhang
e-mail: hyzhang@ucsd.edu

N. Chan
e-mail: nchan@ucsd.edu

C. Hu
e-mail: chh281@ucsd.edu

S. Binnani
e-mail: sbinnani@ucsd.edu

H. I. Christensen
e-mail: hichristensen@ucsd.edu

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_30

by providing on-demand mobility that can address heavy campus traffic during peak hours, while reducing the amount of on-campus parking needed.

Over the course of seven months, AVL has implemented software modules for sensing and estimation, planning, controls, and benchmarking. Additionally, state-of-the-art algorithms for localization, mapping, and perception have been explored and evaluated from multiple open-source communities: Robot OS (ROS) [6] and Autoware.AI. [2]. Although some of those software packages discussed in detail in this paper are robust and work very well for certain applications, they do not generalize and scale, and often require extensive software-hardware integration. The evaluation and development results during the testing phase of the project are presented with proposed modifications to improve performance, flexibility, reliability, and scalability in autonomous vehicle systems for micro-transit.

2 Technical Approach

Each of the two GEM e6 development platforms used for this study has been retrofitted with a drive-by-wire system and a safety disengagement system[1] that provides an interface for controlling steering angle and angular velocity, brake pedal position, acceleration, and gear shifts using ROS. Each of the units is equipped with six Mako G319 cameras, one VLP-16 Velodyne LiDAR, 16 Bosch ultrasonic sensors, and a Varience VMU931 Inertial Measurement Unit (IMU). Additionally, a Watson DMS-SGP02 Inertial Navigation System (INS) is being tested on one of the units. The development platforms are shown in Fig. 1.

Initial system integration and development was performed using numerous tools, including Docker containers for modularization, the open-source Autoware stack as a development platform, alongside communication APIs for vehicle-to-system control. While a reasonable degree of autonomy was achieved during this development stage, Autoware's localization and planning pipeline heavily rely on dense point cloud maps and LiDAR technology. As discussed in later sections, this technology may be adequate for smaller scale environments but comes at a great cost as the size of these maps drastically increase—computationally and maintenance wise. As part of an ongoing process, AVL has started developing alternative methodologies that provide an equal or greater level of system robustness with the primary goal of improving scalability and integration as a whole. In this section, we introduce tools and the current state of the system and propose alternatives for improvement.

---

[1] The disengagement system permits safety drivers to intervene at any time using the steering wheel, pedals, or emergency button. During field tests, steering wheel and pedal interventions proved to be necessary due to the immediate response needed in many scenarios.

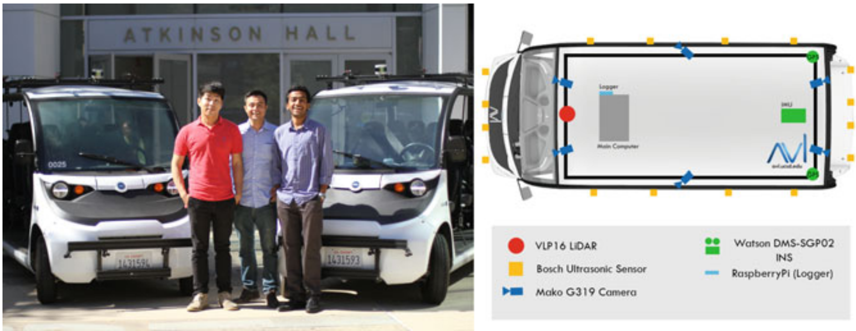


Fig. 1 AVL team members and GEM e6 Development platforms are shown on the left. Sensor locations are shown on the right

2.1 Containers

ROS is an ecosystem that provides a convenient communication interface for our applications and modules. Although ROS offers this convenience, one drawback lies on package interdependencies and libraries (i.e., OpenCV) which often leads to problems compiling source code due to incompatible versions. By isolating the different modules into separate environments using Docker containers, this task becomes trivial and allows us to automate the module instantiation process during boot time. UC San Diego's AVs are among the first few platforms to make use of Docker for robotics applications [10].

2.2 Autoware/ROS

Two of the most popular open-source autonomous driving stacks with ROS support include Baidu's Apollo and Tier IV's Autoware.AI. Ultimately, Autoware.AI was chosen as the foundation of AVL's autonomous driving stack due to its open-source community support and flexibility for modifications and integration. Autoware provides algorithms and packages for perception, mapping, localization, and planning algorithms required for navigation through specific environments. The modules are relatively complete and work well for basic environments, despite some limitations which will be discussed in the latter part of the paper. Several global and local planning modules are also packaged with Autoware, including the A*, lattice, and OpenPlanner, where each planning algorithm is optimized for different scenes.

Out of the different planners provided by Autoware, we chose OpenPlanner [4], which serves as the path planner for autonomous navigation through an urban environment. OpenPlanner is comprised of two distinct modules: a global planner for generating an initial reference path and a local planner for controlling the behavior of the vehicle. Since AVL's focus is on last mile transportation, our efforts have centered around exploration and modification of the local planner. OpenPlanner supports local planning with a limited number of static road features, such as lanes, stop signs, and dynamic features, such as traffic lights. As of Autoware version 1.8, OpenPlanner lacks support for many interactive road features, such as crosswalks and intersections, which are critical for autonomous navigation in areas with high foot traffic—such as UC San Diego. This entails integration of perception modules into OpenPlanner and addition of the appropriate behavior logic to the local planner. In addition to missing local planner logic for certain road features, modifications to the perception modules are required to generalize the detection and tracking algorithms for a wider variety of scenes. In the following sections, we will discuss the specifics of the necessary changes.

2.3 Perception and Tracking

2.3.1 Ground Removal

In the LiDAR detection pipeline, point clouds are projected onto a 2D plane for segmentation and point cloud ground removal to avoid obstacle misclassifications, such as classifying ground points as an obstacle. One proposed implementation is to remove the points that fall below a threshold, or better, to use the Random Sample Consensus (RANSAC) algorithm to fit a plane on the ground. Although these two methods work reasonably well to a certain extent, a failure case is encountered along roads with high degree of inclination as represented in Fig. 2.

To address ground removal along dynamic road conditions, a simple, but effective technique is to partition the point cloud based on the longitudinal distance to the car and fit a plane in each partition. These planes can provide better plane approximations

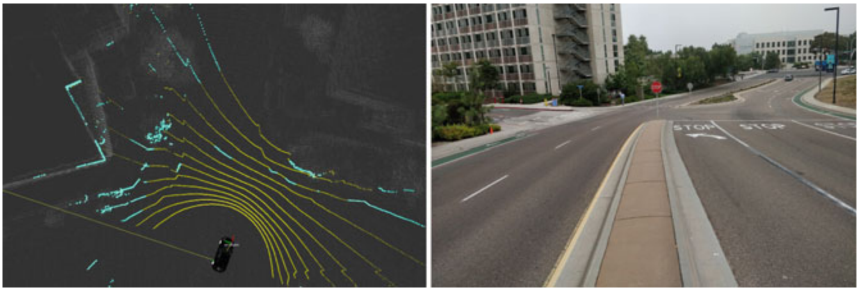


Fig. 2 Ground remove failure case along inclines: in the LiDAR scan image (left), yellow points are classified as ground and turquoise points are classified as obstacles. The vehicle is turning left in a three-way intersection and part of the ground is classified as obstacle. The image on the right shows the intersection

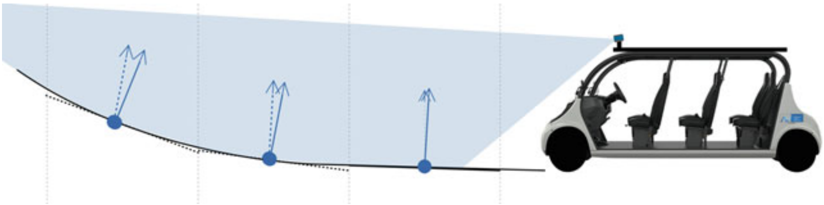


Fig. 3 The direction of dashed arrows indicate the RANSAC initializations and the solid line arrows correspond to the normal vectors of the estimated plane

for curved roads and remove the ground points effectively, as demonstrated in Fig. 3. The RANSAC plane fitting implementation in the Point Cloud Library (PCL) [9] is initialized with a direction that is approximately normal to the plane in question and generates a normal vector that best characterizes and fits the data, i.e., an angle within a threshold. Using this method, an iterative approach is applied to specify the direction of the planes. First, we assume that the plane that is closest to the vehicle is nearly perpendicular to the z-axis in the LiDAR frame and use this direction as input to determine the actual parameters of the plane by RANSAC. Based on the assumption that the slope of the road changes slowly among adjacent partitions, we use the direction of the normal vector of the last estimated plane as the input of the estimation of the next plane.

In Fig. 3, we separate the road into three segments and estimate from right (the closest partition to the vehicle) to left. This approach works effectively even when the road is curved as shown in Fig. 4, but the assumption that each partition can be approximated by a plane might not always hold when the change in slope varies significantly within short distances. A similar approach for iterative elevation calculation is considered in [8].

2.4 Detection and Pose Estimation of Vehicles

For obstacle detection and avoidance, it is essential to know the size, shape, and ultimately the centroid and classification of a detected object with a high degree of reliability. While LiDAR data provides estimates with centimeter level accuracy, it can be challenging to estimate the pose of an object out of a single point cloud. For vehicles, laser scans may reflect off a single side of the vehicles making it challenging to estimate their true width, as shown in Fig. 5. Additionally, the LiDAR vertical field of view and resolution can deteriorate significantly for objects that are further away depending on the number of LiDAR channels.

In the ideal case in which LiDAR data is provided for two adjacent sides of a vehicle, we can extract an L-shaped point cloud when it is projected onto the XY plane. Thus, we can estimate the pose based on this information. Bounding box and orientation fitting methods [14] have been explored, but in this section, we describe

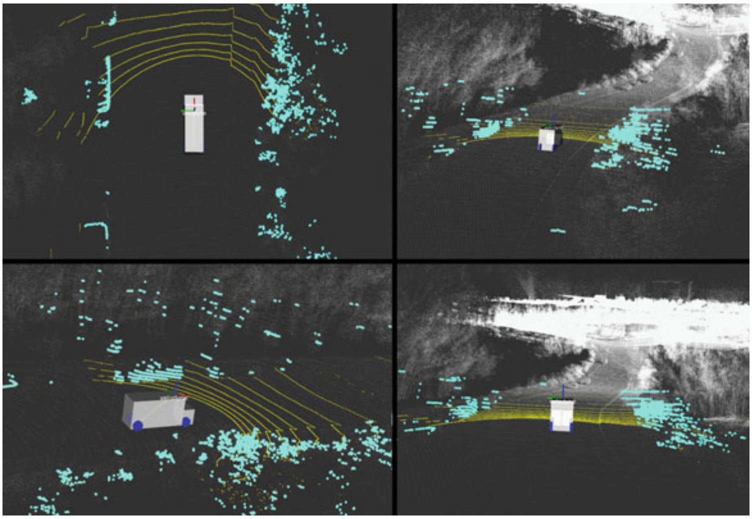
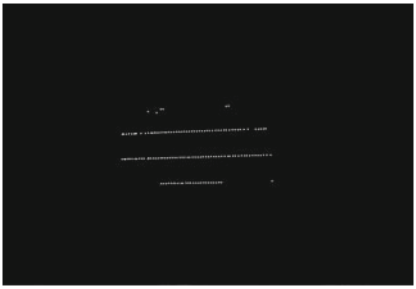


Fig. 4 Successful ground classification along an inclined road from multiple perspectives. The yellow scans are classified as ground and are removed. Turquoise scans are considered as potential obstacles

Fig. 5 16-channel LiDAR scans reflecting on a single side of a vehicle



an alternative RANSAC-based line fitting method that can also be applied for this application.

Based on the assumption that most extracted points define two nearly-orthogonal intersecting planes, we first project our data on a plane and find the first best-fit orthogonal line using a RANSAC approach and remove all the points that are relatively close to this line. The second line is estimated with the remaining points while enforcing the constraint that it should be perpendicular to the first line. Based on the two lines and prior knowledge of vehicle sizes, we approximate and fit a bounding box for the corresponding point cloud, as shown in Fig. 6.

In realistic scenarios, these ideal cases are seldom encountered. During the tests performed, vehicles that are located along the XY-axis, with respect to the LiDAR coordinate frame, project a single side and do not provide enough information to estimate width. Integration of relative position, lane information and even tracking

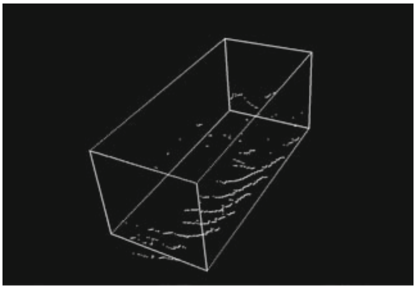


Fig. 6 Bounding box generated using LiDAR-based classification

information is needed for a more robust vehicle classification and tracking system. On the other hand, for obstacle detection and avoidance, it is sufficient to cluster points together to represent an unlabeled obstacle.

2.5 Vehicle Control and Path Tracking

The trajectory tracking algorithm used in Autoware is the geometric Pure Pursuit algorithm, which considers the upcoming waypoint as the desired pose and computes the required linear and angular velocities based on a geometric curve fit. Keeping in mind the environment and the low speeds (a maximum of 25 mph) at which we operate our system inside the university campus, once the reference trajectory and its curvature is considered, we employ a PID strategy for the longitudinal control and a bicycle model of the car for the steering control with an angular velocity that is directly proportional to the current speed of the vehicle. Given that the acceleration and braking capabilities of the car are different due to inherent mechanical constraints, it is also necessary to have different PID controllers for acceleration and braking. Extensive testing was done in this regard to achieve the appropriate constants for acceleration and braking that suited varied road inclines present on campus as shown in Fig. 7.

When coupled with a purely geometric path tracking algorithm, the cornering abilities of the driving are questionable at higher speeds. Cases where the vehicle shot off the path while driving at relatively higher speeds around sharp corners were experienced, which led us to explore other strategies to help tackle this issue. In view of that, the kinematic model for path following [11] was studied and researched upon for the current application. However, after careful analysis of the advantages of the pure kinematic model, it was found to be comparable at lower speeds to that of our current version and without any additional benefits.

As part of an ongoing research, we seek inspiration from one of the major areas of current research in autonomous driving, Lane Keeping Assists Systems (LKAS). A Vision-Based Lane Assist System using deep learning models like DeepLanes [5] and SCNN [13] to estimate the vehicles position and heading in the lane, and thereby

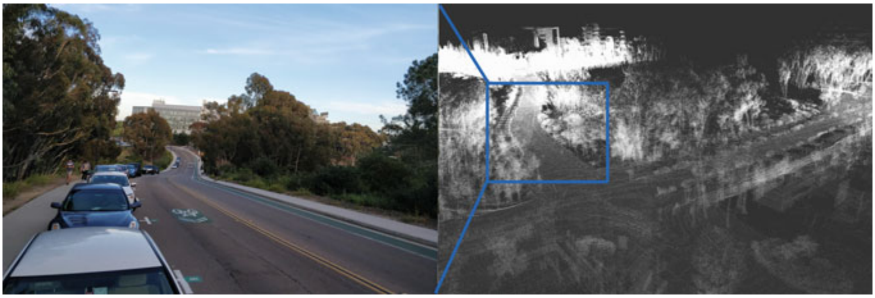


Fig. 7 UC San Diego: Voigt Drive hill shown on the left with the associated point cloud map shown on the right

obtain the cross-track error and desired orientation of the vehicle, is being explored. On the desired trajectory obtained, a Model Predictive Control (MPC) approach is followed as the control strategy. With the reference trajectory coming in from vision-based lane detection, the MPC controller seeks to minimize the cross-track error and the errors in heading direction and velocity of the car. (1) represents the cost the MPC controller is trying to minimize. The constraints that are derived from the kinematic model and those that govern the optimization are shown in (2).

$$J = \Big(\sum_{t=0}^{t=N-2} K_1 e_{cte,t}^2 + K_2 e_{\psi,t}^2 + K_3 (v_t - v_{ref})^2\Big)$$

$$+ \Big(\sum_{t=0}^{t=N-2} K_4 a_t^2 + K_5 \delta_t^2 + K_6 (a_t - a_{t-1})^2 + K_7 \big(\delta_t - \delta_{t-1})^2\big) \tag{1}$$

$$\begin{aligned} &x_t + v_t \cos(\psi_t)\, dt - x_{t+1} = 0 && y_t + v_t \sin(\psi_t)\, dt - y_{t+1} = 0 \\ &v_t + a_t dt - v_{t+1} = 0 && \psi_t + v_t \frac{\delta_t}{L_f} dt - \psi_{t+1} = 0 \\ &f(x_t) - y_t + v_t \sin(e_{\psi,t}) - e_{cte,t} = 0 && \psi_t - \psi_{ref} + v_t \frac{\delta_t}{L_f} dt - e_{\psi,\, t+1} = 0 \end{aligned} \tag{2}$$

x_t, y_t, ψ_t : *pose at time t*
v_t : *velocity at time t*
a_t, δ_t : *acceleration and steer at time t*
$e_{cte,t}$: *cross track error at time t*
$e_{\psi,t}$: *error in heading at time t*
$f(x_t)$: *reference trajectory*
v_{ref} : *reference velocity*
ψ_{ref} : *desired heading*
$K_{1:5}$: *weighting constants*
L_f : *wheelbase of the car*

2.6 AVL Logger

As of January 2019, the California Department of Motor Vehicles has issued 62 Autonomous Vehicle Testing permits to large vehicle manufacturer corporations and start-ups to perform field tests with a safety driver. As part of the California Autonomous Vehicle Testing Regulations, every manufacturer is required to submit an annual report with a summary of system disengagements. Although most publicly available reports include a summary with the number of disengagements over a period of time, there are many important aspects that are missing. Some of these aspects that are not evaluated include Mean Time Between Interventions (MTBI), Mean Distance Between Interventions (MDBI), comparisons between autonomous vs. manual driving, AV impact on power consumption, and long-term implications such as maintenance and overall cost of ownership. These metrics require a more comprehensive methodology for benchmarking a self-driving vehicle and the implications of long-term autonomy.

As part of an ongoing research effort, AVL has developed a system for standardizing these metrics. The logging system shown in Fig. 8. was designed to log various signals for performance and disengagement analysis in a non-intrusive isolated environment. The logging system is based on Flask and the RESTful API on a Raspberry Pi. The signals collected from the autonomous vehicle computing platforms are transferred through HTTP(S) protocol via ethernet and stored in SQLite databases for future offline analysis (Fig. 9). This vehicle API is a lightweight software package (ROS node) which serializes all of the logging information and encapsulates it using JSON objects.

This external logging mechanism is designed to be used continuously without interfering with the main system and can be beneficial for third-party verification by public-key encryption. Some of the metrics defined for the vehicles being tested at UC San Diego are listed on Table 1.

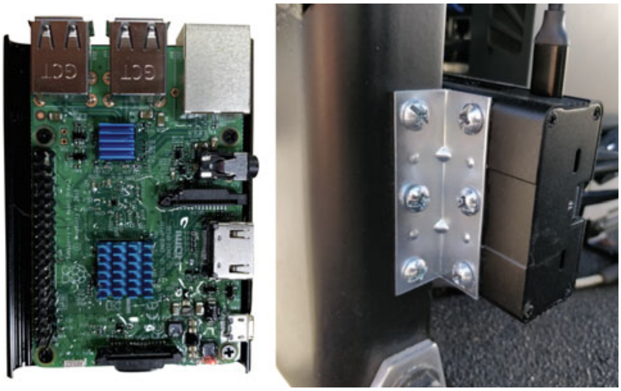


Fig. 8 Raspberry Pi Logging Module mounted on a development platform

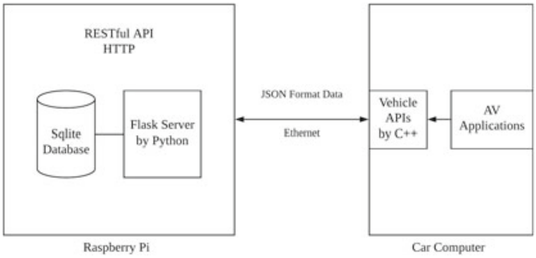


Fig. 9 Logging system architecture

Table 1 Metrics defined for bench marking and evaluation of autonomous vehicles. Typical values for MDBI and MTBI during initial autonomous tests were 0.53 km/intervention and 2.6 min/intervention, respectively. An ongoing study is being conducted to analyze long-term autonomy using these as primary metrics

Trigger	Metric	Type
Intervention	Mean Distance Between Intervention (MDBI)	Event driven
Intervention	Mean Time Between Intervention (MTBI)	Event driven
Energy	Miles per Gallon (MPG) or Charge (mAh)	Continuous
Maintenance cost	Brakes and Tire Wear	Event driven
Up-time	Time Elapsed Per Trip	Continuous
Control	Speed, Acceleration, Steer Angle	Continuous
Location	GPS/Map Coordinates	Continuous

3 Experimental Results

Live tests and data collection are actively performed at UC San Diego for development under different terrain conditions. As seen in Fig. 10, the topological conditions and geometry of the university provide testing grounds in terms of evaluating the robustness of mapping and localization, perception and tracking, motion planning, and logic.

3.1 Localization and Mapping

Extensive localization and mapping tests have been performed at UC San Diego using Normal Distributions Transform (NDT) [1]. The Point Cloud Library (PCL) implementation provides a library for mapping and localization and involves a three-

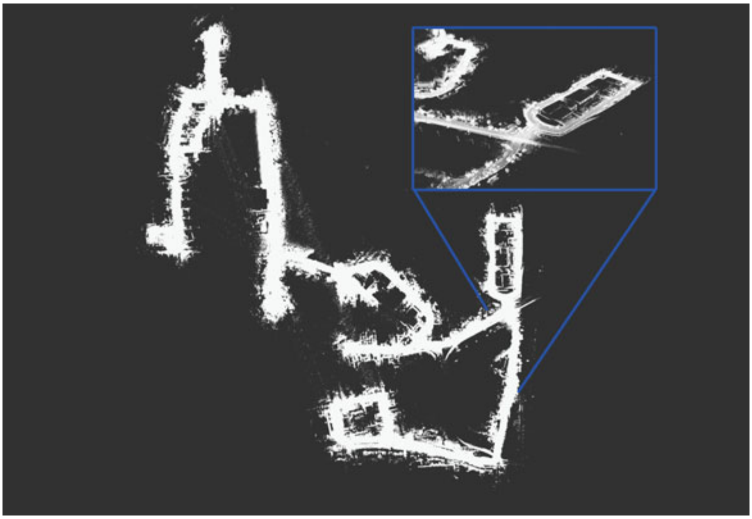


Fig. 10 3D Point cloud map corresponding to mail delivery routes at UC San Diego. The distance of the trajectory is approximately 3.9 miles

step process: data collection, mapping, and post-processing. In the data collection step, a development platform with a sixteen channel LiDAR fixed on the roof of the vehicle was used to collect data by driving along multiple parts of campus. Depending on the size of the data, mapping was performed on a CPU or a GPU. The use of a graphics card can significantly accelerate the process of generating point cloud maps; however, as maps grow large in size, graphics cards begin to run out of internal memory causing the mapping algorithms to crash. This process was tested on a variety of CPUs and GPUs including the Intel Xeon E3-1275, Intel i7-6700, Intel i9-7900, Nvidia Quadro M1000M, Nvidia GTx 1080Ti, and Nvidia Titan Xp.

Figure 10 is an example of a map that is actively used to perform tests on campus. The generation of this map required the use of CPU with a computation time of approximately 28 hours; the total distance along the trajectory is 3.9 miles. In the post-processing step, the map was down sampled using a Voxel Grid Filter approach with $30\,\text{cm}^3$ 3D grids. [9].

The size of the original map generated was 2.8 GB and was reduced to 530 MB after downsampling. During exhaustive testing, it was determined that as point cloud maps exceed 500 GB in size, localization and system performance begins to be impacted significantly and in the worst case (1 GB and above) cause the algorithm to crash or lag long enough to lose the localization prediction. This poses a major challenge in scalability for self-driving stacks that are highly dependent on dense point clouds and LiDAR-based mapping and localization.

Given that NDT employs a stochastic approach, it is robust to certain discrepancies between the original map and changes to the environment but at the same time becomes more complicated to estimate future system failures. Although immediate system failures are detectable by measuring the NDT matching error score, additional sensor data fusion is essential for addressing potential risks that may originate from unexpected behavior—ultimately improving robustness.

3.2 Motion Planning and Logic

The California Department of Motor Vehicles defines an intersection to be a place in which one roadway meets another roadway including cross streets, side streets, alleys, freeway entrances, and any other location where roads join each other that can include unmarked crosswalks [3]. The anticipated mail delivery route at UC San Diego contains 22 crosswalks, some of which are protected by stop signs, and others that require drivers judgement to slow down or stop.

Our crosswalk state implementation is an extension of the OpenPlanner behavior state machine, and contains two additional states: crosswalk approach and crosswalk stop. In the crosswalk approach state, the ego vehicle will reduce its velocity as it becomes within a certain distance the crosswalk. The vehicle will maintain a reduced velocity until it passes the crosswalk and return to its target speed. However, if pedestrians appear on the crosswalk, the vehicle will enter the crosswalk stop state, stop before the crosswalk, and wait for all pedestrians to clear before moving again. A diagram detailing the crosswalk states is shown in Fig. 11.

Addition of the crosswalk class entailed modifications in the OpenPlanner road network class, along with all its supporting functions. However, we note that challenges were encountered during implementation of behavior states in Autoware, primarily due to the proprietary AISAN vector map format. One issue encountered was that crosswalks could only be detected when driving on a single lane, but not along the opposite direction. This stemmed from the fact that each encoded crosswalk feature was linked with a single waypoint, which in turn is part of a single lane. In order to link to all the lanes that the crosswalk intersects, we linked the crosswalk to all lanes that were within a certain radius.

By a similar approach, the foundation for intersection logic is based on the location of all of the stoplines at an intersection; these are encoded in the vector map to determine if a vehicle is waiting at the stop sign or not.

Once the ego vehicle comes to a stop and enters the three second rule required by law, the ego vehicle enters a wait state until an intersection-clear signal is transmitted

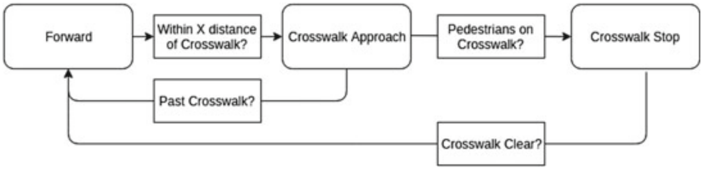


Fig. 11 Introduced crosswalk state in finite state machine

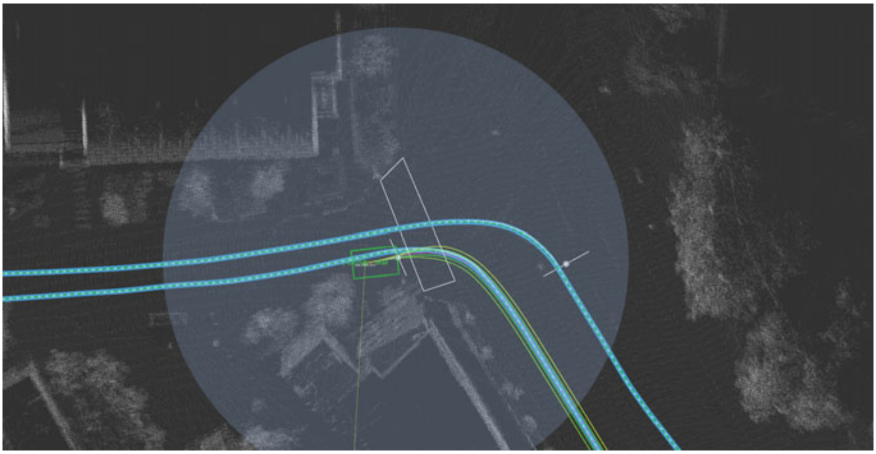


Fig. 12 Simulated ego vehicle reaches a stop line and awaits for intersection-clear signal. Center stop lines are shown by white markers and are used for finding stop lines within a 30 m search radius (shown in blue) from an intersection

that allows it to resume its intended trajectory as shown in Fig. 12. To determine if the ego vehicle is clear to cross an intersection, a vehicle-to-stop sign association must be performed to determine the order in which the other vehicles arrived, if any. Presently, the intersection module is under constant development as perception and tracking is completed. Using a distance metric and camera information, one approach being explored involves a simple plane-to-plane mapping using homographies [12] and real-time vehicle classification data obtained from our four front cameras with a horizontal field of view of approximately 180°.

The current crosswalk and intersection state implementations have been verified via simulation with the modified version of OpenPlanner. The integration of pedestrian detection and intersection logic is ongoing and is expected to be completed in the near future.

4 Conclusion

Numerous lessons were learned during the process of enabling last mile transportation where both pedestrian and automotive traffic are present. The first and foremost concern in last mile transportation is safety, which hinges on accurate vehicle control, LiDAR detection, localization and trained safety drivers.

We surveyed several control algorithms, including PID kinematic control and regular kinematic control. Despite efforts to fine tune the models, both displayed characteristics of overshooting lane boundaries and other hazardous behavior when tested with our development platforms at high speeds. To guarantee safe behavior of autonomous vehicles, an MPC approach, in combination with vision-based lane detection can potentially minimize the caveats we faced with simpler models. We look forward to evaluating this model with our platforms in the future.

While evaluating LiDAR vehicle detection, the assumption of flat ground is insufficient for the hilly terrains encountered at UC San Diego. Since the ground could take up varying portions of the LiDAR points, a ground removal step is necessary for reducing false positives and increasing the accuracy of vehicle detections in a scene. One alternative being explored involves multi-camera and LiDAR fusion with deep neural networks using YOLO3 [7].

LiDAR-based localization and mapping is currently a critical component of autonomous navigation in our implementation of last mile transportation. While the implementation of this technology is feasible, as determined in the course of this study, there are caveats that follow. For micro-transit and smaller scale transportation systems specifically, this technology proposes robust and adequate results given that the maps and vector maps are constantly maintained and updated. On the other hand, as the scale of the environment drastically increases in size and complexity (i.e., small cities or towns), performance is hindered and can result in detrimental errors without additional safeguards in place and extensions that incorporate the notion of context switching between smaller maps. This is an area of research that needs to be addressed for larger scale systems. To address this challenge, AVL is developing a software stack that eliminates the notion of dense point clouds and LiDAR-based localization by introducing a planner that performs under a nominal set of high-level instructions. This approach is intended to handle planning and localization in a similar fashion that human drivers operate with the benefits of 3D situation awareness and dynamic obstacle avoidance.

During this process, we have gained a deeper understanding of open-source state-of-the-art architectures, strengths, and shortcomings for last mile transportation. With our continued focus and efforts to enable autonomous vehicles at UC San Diego, we expect to create a safer environment and more efficient campus transportation.

Acknowledgements We acknowledge the support provided by UC San Diego Logistics Unit, Fleet Services, Police Department, and Mail Delivery Center. We would like to thank Dr. Henrik I. Christensen and Dr. Todd Hylton for making this partnership and project possible, as well Shengye Wang, Dominique Meyer, Eric Lo, Sayan Mondal, Shawn Winston, Francis Joseph, and Ploy Temiyasathit for contributing on different aspects of the project.

References

1. Almqvist, H., Magnusson, M., Kucner, T.P., Lilienthal, A.J.: (2018) Learning to detect misaligned point clouds. J. Field Robot **35**, 662–677 (2018)
2. Autoware: Open-source software for urban autonomous driving. https://github.com/CPFL/Autoware/. Accessed 10 April 2019
3. California Department of Motor Vehicles (2019) California Driver Handbook—Laws and Rules of the Road. https://www.dmv.ca.gov/portal/dmv/detail/pubs/hdbk/right_of_way/. Accessed 10 April 2019

4. Darweesh, H., Takeuchi, E., Takeda, K., Ninomiya, Y., Sijiwo, A., Morales, Y., Akai, N., Tomizawa, T., Kato, S.: Open source integrated planner for autonomous navigation in highly dynamic environments. J. Robot. Mechatron. **29**(4), 668–684 (2017). https://doi.org/10.20965/jrm.2017.p0668
5. Gurghian, A., Koduri, T., Bailur, S.V., Carey, K.J., Murali, V.N.: DeepLanes: end-to-end lane position estimation using deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 38–45 (2016)
6. Quigley, M., et al.: ROS: an open-source Robot Operating System. In: Paper Presented at ICRA Workshop on Open Source Software, Kobe, pp. 12–17 (2009)
7. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement. CoRR (2018)
8. Romanoni, A., Matteucci, M.: Robust moving objects detection in lidar data exploiting visual cues. In: Paper Presented at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, 9–14 October 2016 (2016)
9. Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: Paper Presented at IEEE International Conference on Robotics and Automation, Shanghai, pp. 9–13 (2011)
10. Shengye, W., Christensen, H.I.: TritonBot: first lessons learned from deployment of a long-term autonomy tour guide robot. In: Paper Presented at 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Nanjing (2018)
11. Snider, J.M.: Automatic Steering Methods for Autonomous Automobile Path Tracking. Dissertation, Carnegie Mellon University (2009)
12. Torr, P.H.S., Zisserman, A.: Feature based methods for structure and motion estimation. In: Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, London, 21–22 September 1999 (1999)
13. Xingang, P., et al.: Spatial as deep: spatial CNN for traffic scene understanding. In: Thirty-Second Association for the Advancement of Artificial Intelligence Conference, New Orleans, 2-7 February 2018 (2018)
14. Zhang, X., Xu, W., Dong, C., Dolan, J.M.: Efficient L-shape fitting for vehicle detection using laser scanners. In: Paper Presented at IEEE Intelligent Vehicles Symposium, Redondo Beach, 11–14 June 2017 (2017)

Smooth Local Planning Incorporating Steering Constraints

Freya Fleckenstein, Wera Winterhalter, Christian Dornhege, Cédric Pradalier, and Wolfram Burgard

1 Introduction

Mobile autonomous robots have become increasingly important in agricultural and industrial environments. Compared to standard equipment such as tractors, they have the advantage of being more lightweight thus reducing soil compression and being more flexible thus providing the potential to lower the utilization of chemicals. However, the variability of the crop production requires an appropriate flexibility of the robots with respect to wheel adjustments, such as with the robot depicted in Fig. 1.

The core capability of the navigation system is to generate and execute a collision-free trajectory. A common approach is to use a global planner to compute a path consisting of a list of poses the robot should traverse. In addition, a local planner is used to generate motions connecting through these waypoints. Usual requirements are that the motions are smooth, safe and can accurately be executed even at the via-points. The robot then has to translate the motions into concrete steering commands resulting in the proper steering angles of the wheels and the wheel velocities. Our goal in this paper is to create such a local planner that uses all available degrees of freedom of a robot with individually steerable wheels to effectively navigate while still accurately tracking the global path.

F. Fleckenstein (✉) · W. Winterhalter · C. Dornhege · W. Burgard
University of Freiburg, Freiburg, Germany
e-mail: fleckenf@tf.uni-freiburg.de

W. Winterhalter
e-mail: winterhw@tf.uni-freiburg.de

C. Dornhege
e-mail: dornhege@tf.uni-freiburg.de

W. Burgard
e-mail: burgard@tf.uni-freiburg.de

C. Pradalier
GeorgiaTech Lorraine-UMI, 2958 GeorgiaTech-CNRS, Metz , France
e-mail: cedric.pradalier@georgiatech-metz.fr

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings
in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_31

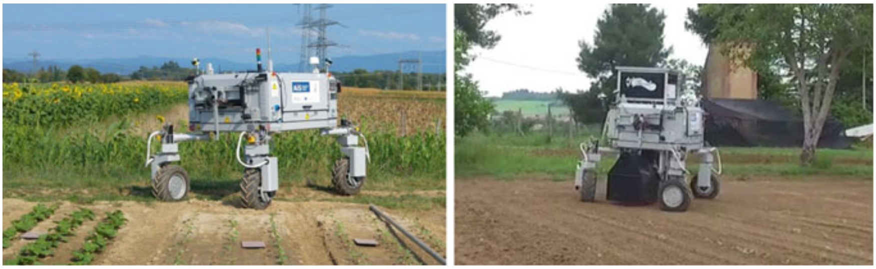


Fig. 1 These pictures show the agricultural robot BoniRob with four wheels whose orientation and rotation can be changed independently. The picture on the right has been taken during an autonomous run

In general, there are several hardware limitations that impose constraints on the available control commands. First, there are the overall velocity and acceleration limits. Second, there are steering constraints such as maximum steering angle and steering velocity, i.e., how fast the wheel orientations can be changed. Our algorithm considers all of these constraints while at the same time generating velocities that lead to high tracking accuracy and efficient overall movements of the vehicle.

The local planner we present is based on the principle of trajectory rollouts [7]. It determines a set of candidate velocities close to the current velocity. It then checks for each candidate whether it satisfies the hardware acceleration and velocity constraints and computes a sequence of velocities to reach the desired velocity from the current one. Finally, it computes a score for each sequence and chooses the best one.

The particular challenge when considering limits on steering angles and steering velocities is that smooth transitions in the velocity of the robot do not necessarily lead to smooth transitions in the wheel angles. For example, to first execute a small forward velocity and then a small sideways velocity, the wheels always need to turn by 90° independent of how small the absolute velocity change is. Therefore, it is necessary to consider changes in the wheel angles explicitly. In this paper, we present two methods to incorporate steering constraints in the local planner by transforming the wheel configurations into the space of the instantaneous center of rotation (ICR). First, we introduce the constraints in a post-processing filter step on the velocity command that was computed without regard for steering constraints. The second approach is to directly incorporate the steering constraints as ICR constraints in the local planner. To achieve this, we compute a local path of valid ICRs toward a candidate velocity and if such a path exists instantiate the ICRs with velocities to gain a trajectory. In addition, we demonstrate how the steering mechanical constraints can be efficiently expressed in a unified manner.

We evaluate both methods on three trajectories posing different challenges in simulation and verify the observed behavior in real-world experiments. The results show that including the ICR constraints with our method greatly reduces the travel time. At the same time, integrating the constraints directly in the local planner maintains high tracking accuracy.

2 Related Work

Many approaches have been presented that compute velocity commands for tracking a given path. Early methods use a set forward velocity and only control the steering to stay close to the designated trajectory [11]. They were, for example, successfully used in controlling vehicles on a motorway [9]. Similarly, several approaches have been developed that are purely reactive and directly map sensor data to steering corrections [2]. These approaches are not sufficient for the problem we consider, as they might lead to great changes in required steering angles. Other methods restrict the possible paths to be twice differentiable [3]. Our experiments reveal that we do not need to comply with this restriction to get a low tracking error.

Some higher level approaches use model predictive tracking strategies to estimate the future robot state [5, 13]. Fox et al. and Gerkey and Konolige [6, 7] sample a set of velocities, predict the robot pose one timestep into the future, compute a score for each velocity, and execute the best. Our approach is similar in that we also sample a set of velocities and score them.

There are two major differences to our approach. First, we propose to generate sequences of velocities toward a sampled velocity and include a path prediction in the score. This allows us to plan further ahead and create more stable sequences. In particular, mechanical constraints may require the robot to slow down, which we can incorporate. The pose paths resulting from these velocity sequences are similar to the motion primitives in state lattice planning [12]. Second, we consider mechanical steering constraints in the velocity sequence generation. While most local planner methods incorporate velocity and acceleration limits [6, 10], few of them consider these additional constraints. For integrating the mechanical steering constraints, we leverage previous work [4, 14]. Clavien et al. show that the velocity of a ground vehicle corresponds directly to the ICR coordinates expressed in homogeneous coordinates that can be mapped onto a unit sphere in 3D [4]. The advantage of the spherical coordinates is that they are bounded and that any transition between two ICRs can be represented by a segment of a great circle on the sphere. If the segment crosses the equator, then the robot wheels will temporarily be aligned and the path curvature will change sign at this point. Schwesinger et al. demonstrate that steering velocity constraints can be expressed by half-plane constraints on the ICR displacement [14]. In this paper, we combine these results and demonstrate how wheel mechanical constraints can be expressed efficiently in a common frame.

3 Integrating Mechanical Steering Constraints

In this section, we demonstrate how the steering mechanical constraints can be efficiently expressed in a unified manner. We consider three constraints: the maximum steering velocity, the distance to a wheel, and the rotation limit of the steering joints. In a practical deployment, having an ICR too close to one of the vehicle wheels leads to mechanical instabilities: a small numerical variation in the ICR least-square position estimate [4] will result in large steering angle changes, which will put a continuous strain on the wheel. Additionally, few vehicles have steering joints that can rotate continuously. E.g., in our system, it is important to avoid moving from steering angle 89° to 91°, as the wheel will have to be rotated by 178°.

3.1 Formalizing Mechanical Steering Constraints

We describe the mechanical steering constraints in ICR coordinates expressed in the robot coordinate frame. Consider the homogeneous coordinates of an ICR as 3D points. Then, any point P of the homogeneous plane is projected orthogonally on the surface of the unit sphere centered at (0, 0, 0) by normalizing it as $\Pi(P)$. Points at infinity (i.e., straight-line motions) project to the equator of the sphere. The north pole of the sphere corresponds to an ICR representing rotation on the spot. Inversely, from a point (u, v, w) on the sphere, the corresponding ICR can be found as $(u/w, v/w)$ if $w \neq 0$. As a result, diametrically opposite points are equivalent. For any ICR I in homogeneous coordinates, $\Pi^+(I) = I/\|I\|$ and $\Pi^-(I) = -I/\|I\|$ are its equivalent diametrically opposed projections on the sphere. We define $\Pi(I)$ as the tuple $\left(\Pi^-(I), \Pi^+(I)\right)$. Then, a segment $J = [I_1, I_2]$ in the homogeneous space projects to two great circle segments $\Pi(J) = \left(\left[\Pi^-(I_1), \Pi^-(I_2)\right], \left[\Pi^+(I_1), \Pi^+(I_2)\right]\right)$.

We consider a vehicle with wheels W_i, represented in homogeneous coordinates $(x_i, y_i, 1)$. According to our notation, the line joining the i-th wheel to I_1 is $L_i = I_1 \times W_i$. In projective geometry, the cross product of two points is the homogeneous representation of the line going through them, even for points at infinity. Note that a line defined by $ax + by + c = 0$ can be represented in homogeneous coordinates $L_i = (a, b, c)$. Hence, the wheel steering is obtained by $\phi_i = \text{atan2}(b, a)$.

Steering Velocity Constraints Using the results from Schwesinger et al. [14], we express the steering velocity constraints by half-plane constraints on the ICR displacement. Because of the maximum steering velocity, a wheel steering ϕ can change by at most $\pm\delta\phi$ in one timestep. Geometrically, this means that every wheel defines a cone of $\pm\delta\phi$ with boundaries L_i^+ and L_i^- resulting from $W_i \times (-\sin(\phi_i \pm \delta\phi), \cos(\phi_i \pm \delta\phi), 0)^T$. The intersection of the cones defined by all the wheels is the set of feasible ICR changes within one timestep. In the $\mathbb{R}^2$ plane, this intersection is a Nef polygon [1, 8], i.e., a polygon that is defined by the intersection of half-planes and may include points at infinity. The ICR movements and constraints are illustrated in Fig. 2. Assuming the ICR needs to move from I_1 to I_2, let us define the line $\mathscr{L} = I_1 \times I_2$ on which the ICR has to move to achieve the requested ICR. The intersection of $\mathscr{L}$ with the cone defined by L_i^+ and L_i^- is the generalized segment $\mathscr{M}_i = [M_i^-, M_i^+]$. A feasible ICR will hence have to belong to the intersection of the n generalized segments $\mathscr{M}_i$ due to all the wheel constraints, or the equivalent great circle segments $\Pi\left(\mathscr{M}_i\right)$ on the sphere.

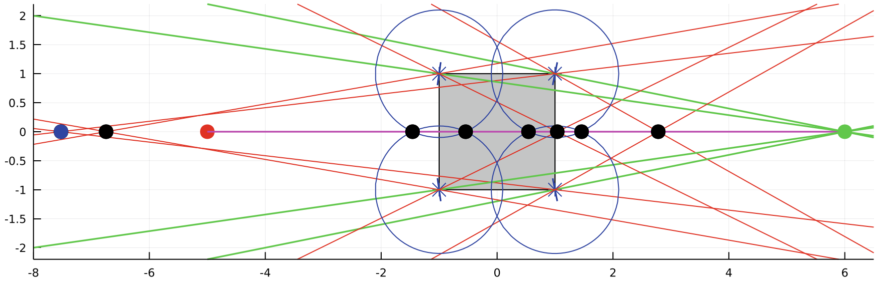


Fig. 2 ICR movement constraints. The robot is depicted in gray, with four wheels. The ICR is initially on the green disk, linked by the green lines to the four wheels, and is requested to move to the red position. The magenta line highlights the ICR path, which crosses the steering velocity constraints in red and the wheel distance constraints in blue. The black dots depict the intersections, and the blue disk is the furthest the ICR can move in one timestep

Wheel Distance Constraints If I_1 and I_2 are not both at infinity, we want to forbid the ICR from moving within a radius r from wheel i. This requires computing the intersection C_i^+ and C_i^- of $\mathscr{L}$ with the circle centered on W_i with radius r. This is solved with basic geometry in $\mathbb{R}^2$, without resorting to projective geometry tricks. Contrary to the steering velocity constraints, the wheel distance constraints are exclusive, so the ICR must be outside the $\mathscr{C}_i = [C_i^-, C_i^+]$ generalized segment.

Rotation Limit Constraint of Steering Joints The wheel mechanical constraints define a specific range of steering that the ICR should not enter. Similar to the steering velocity constraints, this defines a set of exclusion cones in the ICR space. The intersection of $\mathscr{L}$ with these cones gives a set of generalized segments $\mathcal{N}_i = [N_i^-, N_i^+]$ out of which the ICR should always be.

3.2 Constraint Unification

All the above ICR constraints have been expressed as inclusion or exclusion segments for the ICR, so the next challenge is to fuse all these segments efficiently. To this end, we use the process depicted in Fig. 3: we will project all of them on the unit sphere. In practice, all the segment ends have been computed numerically, so we compute the great circle containing all of them using total least squares. Now, we can represent each ICR P with the coordinates $(x, y, 0)$ of its projection on the great circle, or more efficiently by its angular position $\theta^+(P) = \operatorname{atan2}(y, x)$ which is equivalent to the opposite angle $\theta^-(P) = \operatorname{atan2}(-y, -x)$. In summary, every generalized segment $[p, q]$ in homogeneous coordinates projects on a tuple of great circle segments $\Pi\left([p, q]\right)$, which is expressed as an angle interval union

$$\theta\left([p, q]\right) = \left[\theta^-\left(p\right), \theta^-\left(q\right)\right] \cup \left[\theta^+\left(p\right), \theta^+\left(q\right)\right]$$

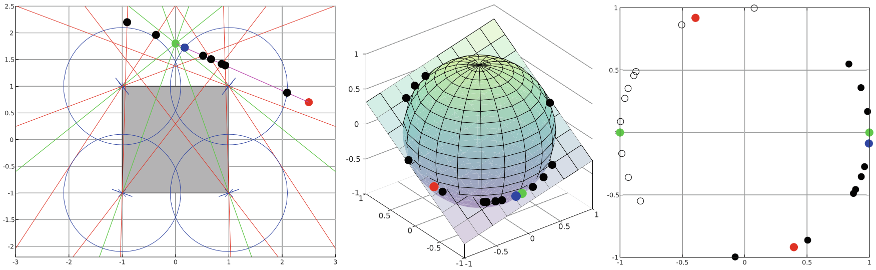


Fig. 3 ICR constraint unification process. Left: the ICR initial and requested configuration in 2D, and the best feasible ICR in blue. Center: projection of the ICR and constraints on the unit sphere, and extraction of the great circle containing all the projections. Right: the ICR constraint on the great circle $\theta(.)$. Black and white disks correspond to opposed constraints

and the challenge is now to intersect a set of intervals in the most efficient way.

We choose to represent a collection of angle intervals by a sorted list of endpoints and a Boolean flag stating if $\pm\pi$ is included or not in the collection. If it is, the first value in the list is the end of the first interval, otherwise, it is the start. This representation has a quick complement operator by toggling the flag as well as a quick union operator which just fuses the sorted list in linear time. The intersection is computed using the complement of the union of the complement.

The unified constraints is the intersection $\mathscr{I}$ of all the constraints:

$$\mathscr{I} = \bigcap_{i=1}^{p} \theta\left(\mathscr{M}_i\right) \cap \overline{\theta\left(\mathscr{C}_i\right)} \cap \overline{\theta\left(\mathscr{N}_i\right)} \tag{1}$$

If $\mathscr{I}$ is empty, then the current ICR is too close to the wheels or the mechanical limits and there is no feasible movement to reach a suitable ICR within one timestep. In general, $\mathscr{I}$ will be a collection of intervals, out of which we select the one containing 0, i.e., $\theta(I_1)$. If it also contains $\theta(I_2)$, then I_2 is feasible and safe. Otherwise, the interval endpoint closest to $\theta(I_2)$ is selected. If $\mathscr{I}$ does not include 0, then the current ICR is not acceptable, but there is a feasible ICR at reachable distance. In this case, the selected interval is the one closest to 0. From the optimal value of $\theta(I^*)$, we can easily recover $\Pi(I^*)$ and a safe ICR I^*. In practice, we can run this process in less than 10 μs, which is a critical requirement to integrate it into the trajectory rollout presented in Sect. 4.

4 From Paths to Velocity Commands

To follow a path, the robot executes velocity commands $v = (v_x, v_y, \omega)$. At each timestep, we determine a set of candidate velocities. For each candidate, we compute a trajectory rollout of discretized velocities and poses stretching over a constant rollout time. Then, we compute a cost for each rollout and choose the best.

Trajectory Rollouts We define a fixed discretization of the velocity space and compute a set of candidate velocities around the current velocity that satisfy the velocity and acceleration constraints of the robot. Assuming maximum acceleration toward the candidate velocity in v_x, v_y and ω, a velocity rollout is generated. If the candidate velocity is reached before the rollout time ends, this velocity is kept for the remaining time. The pose rollout is then computed by successively applying the velocities from the velocity rollout with a given timestep to the current pose.

Velocity Cost During path execution, the robot has to trade-off efficiency and accuracy. For example, in a 90° turn, the most accurate execution would be going straight, stopping, turning, and going straight. In contrast, the more efficient, but less accurate solution would be to cut the corner in favor of a smoother and faster execution. As a measure of accuracy, we use the integrated distance $d(P, T)$ of the trajectory rollout T to the path P. For each point in the trajectory rollout, we determine the closest point on the path and compute the distance between those two points. We approximate the integrated distance by the sum over these distances. We model the efficiency by the length of the covered path segment $l(P, T)$. The covered path segment is defined by the first and last trajectory rollout matches onto the path. We balance the effects of these two costs by introducing the *path length scale* $s \in [0, 1]$. The velocity cost is then given as $c(P, T) = (1 - s) \cdot d(P, T) + s \cdot l(P, T)$. For larger values of s, the length of the covered path segment gains more weight so that the local planner will prefer rollouts that cut corners.

5 Integrating Steering Constraints into the Local Planner

While the approach described above takes into account low-level hardware constraints, it does not consider mechanical steering constraints. As we explained above, these constraints can be expressed in the ICR space in a unified manner. To exploit this model of the steering constraints, we project the velocities (v_x, v_y, ω) into the ICR space expressed in homogeneous coordinates $(-v_y, v_x, \omega)$ [4]. If $\omega \neq 0$, the ICR coordinates in the $\mathbb{R}^2$ plane are $(-v_y/\omega, v_x/\omega)$. Otherwise, the robot is moving straight, which corresponds to an ICR at infinity in the direction $(-v_y, v_x)$, which is exactly what homogeneous coordinates $(-v_y, v_x, 0)$ express. There are two possibilities to integrate ICR constraints into the velocity command execution pipeline. First, the ICR constraints can be enforced in a post-processing *Filter* step. Second, the ICR constraints can be integrated directly into the trajectory rollout computation.

5.1 Mechanical Steering Constraints in a Post-processing Filter

To filter the velocity commands in a post-processing step, we first compute the trajectory rollouts in the local planner as described in Sect. 4. The filter then receives the requested velocity command from the local planner as (v_x^r, v_y^r, ω^r). We compute the corresponding ICR $I^r = (-v_y^r, v_x^r, \omega^r)$. Together with the current ICR I^c, we find an acceptable ICR $I^* = (i_x^*, i_y^*, w^*)$ that satisfies the mechanical steering constraints and is as close to I^r as possible. We then compute the filtered velocity command $v^* = (v_x^*, v_y^*, \omega^*)$ by projecting I^* back into the velocity space.

However, an ICR is equivalent to a velocity only up to a scale factor. The last stage is to scale the ICR to output velocities as close as possible to the requested command. Two cases must be considered: If $\|(-i_y^*, i_x^*)\| = 0$, then we set $\omega^* = \omega^r$ and $v_x^* = v_y^* = 0$. Otherwise, we scale $(-i_y^*, i_x^*, w^*)$ with a factor α to give $(v_x^*, v_y^*) = \alpha(-i_y^*, i_x^*)$ the same norm and direction as (v_x^r, v_y^r), which means keeping $v_x^r v_x^* + v_y^r v_y^* \geq 0$. This yields a velocity that satisfies the mechanical steering constraints with the same translational velocity as the requested command.

5.2 Mechanical Steering Constraints in the Local Planner

To integrate the steering constraints directly into the local planner, these constraints have to be considered when generating candidate velocities as well as during trajectory rollout computation. First, we discard candidate velocities that do not satisfy the constraints in the ICR space. A more complex problem is generating trajectory rollouts that are compliant with the steering constraints.

Given the current ICR I^c computed from the current wheel configuration and the candidate ICR I of the candidate velocity v, we compute an ICR path with a limited steering angle change in each timestep. If the candidate ICR is not reachable within the rollout time, the candidate velocity is discarded. Given the current velocity $v^c = (v_x^c, v_y^c, \omega^c)$, the candidate velocity $v = (v_x, v_y, \omega)$ and an ICR path $\{I^0 = I^c, I^1, \dots, I^n = I\}$, we compute a velocity rollout as follows:

For the previous velocity v^{i-1} and the required ICR $I^i = (i_x, i_y, w)$ solve

$$\begin{aligned} \arg\min_\alpha \|\alpha \cdot (i_y, -i_x, w) - v\| \text{ s.t. } & \alpha \cdot i_y \in (v_x^{i-1} - a_x^{\max} \cdot t, v_x^{i-1} + a_x^{\max} \cdot t) \\ & \alpha \cdot -i_x \in \left(v_y^{i-1} - a_y^{\max} \cdot t, v_y^{i-1} + a_y^{\max} \cdot t\right) \\ & \alpha \cdot w \in \left(\omega^{i-1} - a_\omega^{\max} \cdot t, \omega^{i-1} + a_\omega^{\max} \cdot t\right) \end{aligned}$$

where $a_x^{\max}, a_y^{\max}, a_\omega^{\max}$ are the maximum accelerations and t is the length of one timestep. The acceleration limits are set according to the hardware limits of the robot, and the timestep is set to the internal control cycle of the firmware. Intuitively, this provides us with a velocity that is as close to the candidate velocity as possible, while being within the acceleration limits and compatible with the next ICR in the ICR path. This yields a velocity rollout with the additional constraint that the steering angle does not change by more than a given threshold in one timestep. The pose rollout is computed from the velocity rollout as before.

6 Evaluation

We evaluated our approach in simulated and real-world experiments to show how an ICR-aware local planner affects the navigation of the robot and which way of integrating ICRs performs better. In particular, we investigate if considering ICRs leads to higher efficiency during path execution as the robot does not have to stop to turn the wheels. At the same time, we investigate how much the accuracy is impacted. To this end, we considered three different paths to follow: First, a path of smoothly connected *lines and arcs*; second, a *field*-like path with long straight lines and tight 180° turns; and third, a *rectangular wave* path with varying side length, shown in Figs. 4, 5, and 8. All experiments were conducted with maximum velocities set to $v_x^{\max} = v_y^{\max} = 0.2\,\text{m/s}$, $\omega^{\max} = 0.1\,\text{rad/s}$ and $v_x^{\max} = v_y^{\max} = 0.4\,\text{m/s}$, $\omega^{\max} = 0.2\,\text{rad/s}$, respectively. We will reference these two settings as maximum velocity 0.2 and 0.4 in this section. We additionally modified the influence of the covered path length on the velocity rating, using path length scales of 0.08, 0.1, and 0.12. All experiments were performed in simulation with three repetitions. In addition, we evaluated the field path with 0.4 maximum velocity and a path length scale of 0.08 in a real-world scenario on our BoniRob robot shown in Fig. 1 to verify the results

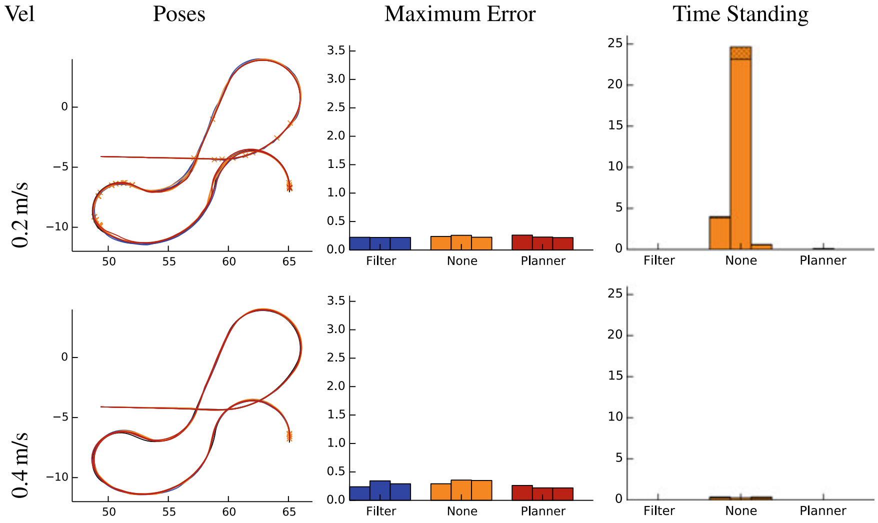


Fig. 4 The results for the lines and arcs path with path length scale 0.12

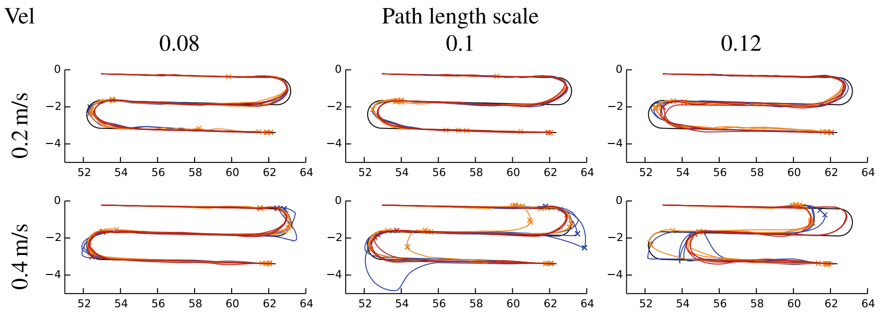


Fig. 5 The evaluated trajectories for the field path. Stops are marked with x

Table 1 The mean execution times for path length scale 0.08 for all trajectories and all evaluated algorithms with different velocities

	Maximum velocity 0.2			Maximum velocity 0.4		
	Filter (s)	None (s)	Planner (s)	Filter (s)	None (s)	Planner (s)
Lines and arcs	385	492	383	182	214	182
Field	169	257	152	147	192	122
Rectangular Wave	235	324	221	292	374	302

of the simulation experiments. We used the odometry to determine the pose of the robot.

We analyze several criteria that are relevant for the local planner performance: A plot of the path to follow in comparison to the executed trajectory to judge how closely the robot is able to follow a given path; the tracking accuracy along the path; the overall mean execution time and, to gain a closer insight on the last point, the time spent standing to perform wheel orientation changes. As a measure for the tracking accuracy, we use the maximum error between the path and the executed trajectory. We show experiment results for our approach with different variants of how to include ICR constraints: including the ICR constraints directly in the trajectory rollouts of the local planner yielding an ICR-aware local planner (Planner), no ICR-awareness (None), and without ICR-awareness but an additional filter run between the local planner output and the command execution (Filter).

First of all, Table 1 clearly shows that considering the necessary wheel angle changes notably reduces the execution time, both with a post-processing filter and when incorporating the constraints directly into the planner. The approach None is always slower than both other variants, by up to a factor of 1.7 (Field, maximum velocity 0.2). Shown are the results for path length scale 0.08; For other path length scales the results are similar.

The lines and arcs path does not pose a challenge to any of the algorithm variants as it only contains straight lines and wide arcs and thus does not force the robot

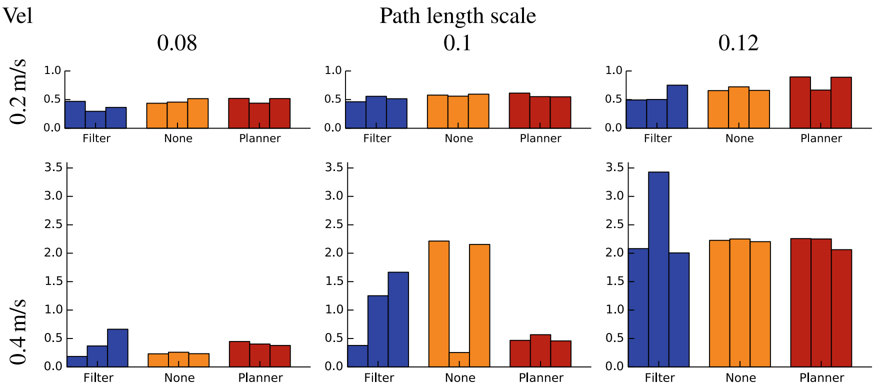


Fig. 6 The maximum error for the field path

toward its steering constraints. Figure 4 shows the results for path length scale 0.12, for other path length scales the results are similar. It is already visible in the trajectory overlay that all variants have a good tracking accuracy. This is confirmed by an overall maximum tracking error of about 35 cm. Neither the filter nor the ICR aware planner spent any considerable time standing, while the planner without regard for its wheel configurations does. All in all, these results demonstrate that on a smooth path with wide arcs, it is beneficial to consider the wheel configuration, but it suffices to avoid unnecessary wheel changes in a post-processing Filter.

The field path contains tight 180° turns. Here, the influence of the path length scale becomes more visible as it allows the robot to shortcut the turns (Fig. 5). Nevertheless, one might choose a larger path length scale to allow the robot to perform smoother curves, thus reducing standing times. This effect is independent of considering ICR constraints. Figure 6 shows that the maximum error increases when increasing the path length scale, especially for a maximum velocity of 0.4. In Fig. 7, we see an explanation for the overall higher efficiency of the ICR aware variants Filter and Planner: They both have notably smaller standing times than without considering ICRs in None. The ICR aware variants are able to use the increased path length scale and show reduced standing times almost down to zero. This is not the case for None with the larger maximum velocity of 0.4 as it does not consider steering constraints and thus cannot use the increased path length scale to prevent excessive wheel orientation changes.

In Fig. 5, we also see that the Filter overshoots the turns on multiple occasions as it prevents turning the wheels to stay within the steering constraints when it would be necessary to do so for the turn. On the one hand, this is an undesirable behavior that neither the Planner nor the None variant faces. On the other hand, Table 1 shows that the Filter succeeds in increasing efficiency. While similar to the Filter and None, the Planner is also affected by the reduced accuracy when increasing the path length scale, it neither shows overshoots as the Filter nor does it take as long as None. Due to the shortcuts caused by the path length scale, effects on the accuracy due to

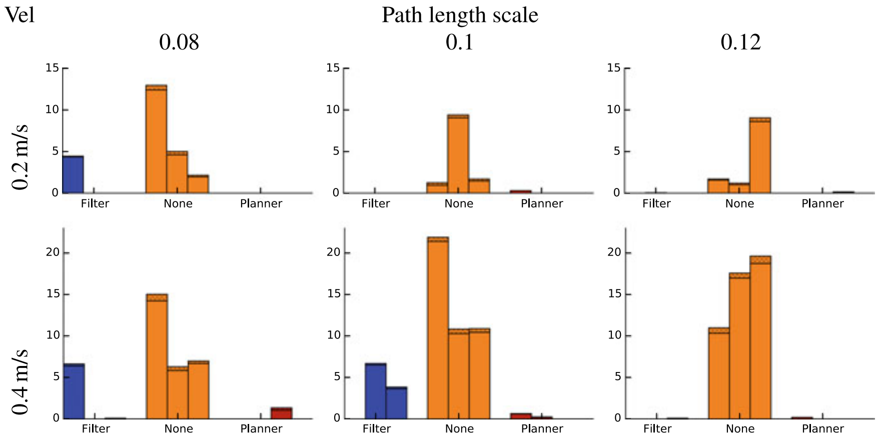


Fig. 7 The total time where the robot was standing in field. Times where the robot was standing, but not turning the wheels are shaded with x

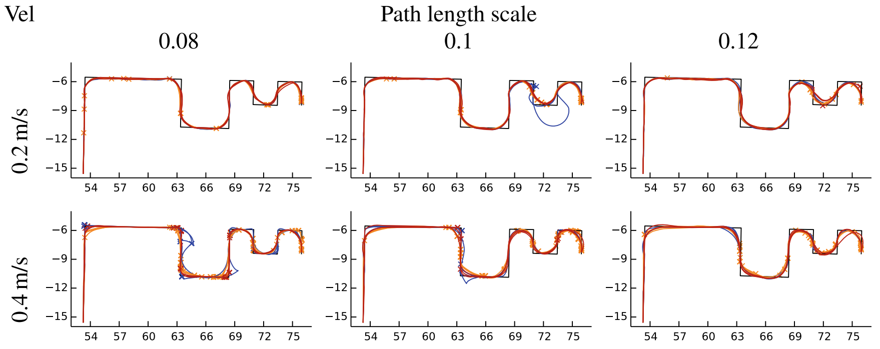


Fig. 8 The evaluated trajectories for the rectangular wave path. Stops are marked with x

considering ICRs are not clearly visible. It is still a relevant case as it directly maps to an application in agricultural robotics.

The third experiment on the rectangular wave path poses similar challenges of tight 90° turns, but does not provoke shortcuts as the field path does (see Fig. 8 for the trajectory overlays). Therefore, we did not observe any strong relationship of the path length scale to the accuracy. While all variants appear to be mostly close to the path with some shortcuts, the Filter clearly drifts away from the path in some cases. In this path, one reason for the increased efficiency when considering steering angles becomes apparent in Fig. 9. While a larger path length scale does reduce the standing times for each variant, being aware of the ICRs pays off notably as the None variant keeps the robot standing the most. In contrast, in most cases, the Filter and Planner show similarly small standing times. The exception is the combination of the smallest path length scale (0.08) with the larger maximum velocity of 0.4

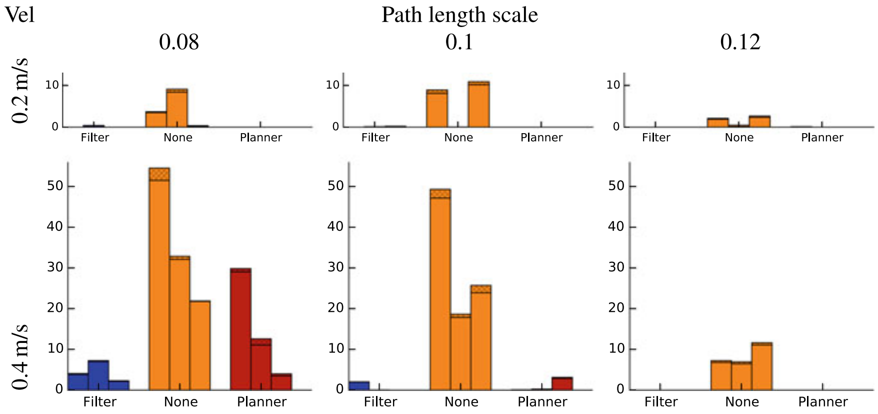


Fig. 9 The total time where the robot was standing in rectangular wave. Times where the robot was standing, but not turning its wheels are shaded with x

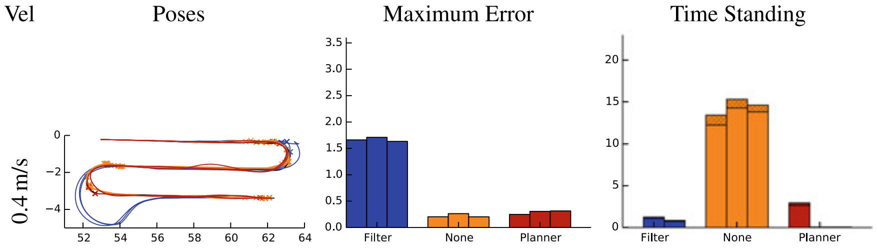


Fig. 10 The results for the real-world experiment

forcing the robot in tight turns. While still better than None, the planner keeps the robot standing more than Filter for two reasons. First, the Planner keeps the robot standing when necessary, i.e., in the turns, while None also stands on straight lines. Second, the Filter only achieves the lower standing times than Planner at the cost of overshooting the path multiple times. The Filter provides a trade-off tailored toward smooth driving at the cost of following the path accurately.

The real-world experiment shown in Fig. 10 shows this even more clearly. Based on the results of the simulation experiments, we set a path length scale of 0.08 for the field path and a maximum velocity of 0.4. The comparison of the three variants on the real robot confirms the results from the simulated data. Filtering the velocity commands using ICR constraints helps to drastically reduce the time spent standing and thus increases efficiency, but leads to large tracking errors mainly when it overshoots the path to keep execution smooth. In contrast, including the ICR constraints in the local planner also reduces the time spent standing, but retains the tracking accuracy of the local planner that does not consider ICR constraints and thus provides the best combination of both options.

This is an observation that we make throughout all four experiments: Considering ICR constraints is necessary to prevent the robot from standing to perform wheel reconfigurations. As the Filter always prevents excessive wheel rotations, it sometimes grossly overshoots the path. The planner is able to make this trade-off of preventing unnecessary wheel movements, while at the same time allowing accurate path tracking.

7 Conclusions

In this paper, we presented novel approaches based on trajectory rollouts for a local planner that computes velocity commands for robots with constraints on steering angles and steering velocities. In extensive experiments, we demonstrated that our approach leads to higher efficiency by minimizing time spent for wheel reconfigurations. At the same time, it is able to highly accurately track the planned path when integrating the constraints directly in the planner.

Despite these positive results, we still see room for further improvement. In our future work, we will investigate computationally feasible methods that estimate the true error between path and rollouts more precisely to further increase the tracking accuracy.

References

1. Bieri, H.: Nef polyhedra: a brief introduction. In: Geometric Modelling, pp. 43–60. Springer (1995)
2. Björn, Å., Baerveldt, A.-J.: A vision based row-following system for agricultural field machinery. Mechatronics **15**(2), 251–269 (2005)
3. Cherubini, A., Chaumette, F., Oriolo, G.: Visual servoing for path reaching with nonholonomic robots. Robotica (2011)
4. Clavien, L., Lauria, M., Michaud, F.: Estimation of the instantaneous centre of rotation with nonholonomic omnidirectional mobile robots. Robot. Auton. Syst. **106**, 47–57 (2018)
5. Conceição, A.S., Moreira, A.P., Costa, P.J.: A nonlinear model predictive control strategy for trajectory tracking of a four-wheeled omnidirectional mobile robot. Optim. Control Appl. Methods **29**(5), 335–352 (2008)
6. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**(1), 23–33 (1997)
7. Gerkey, B.P., Konolige, K.: Planning and control in unstructured terrain. In: Workshop on Path Planning on Costmaps, IEEE International Conference on Robotics and Automation (ICRA) (2008)
8. Hachenberger, P., Kettner, L., Mehlhorn, K.: Boolean operations on 3d selective nef complexes: Data structure, algorithms, optimized implementation and experiments. Comput. Geom. **38**(1–2), 64–99 (2007)
9. Jurie, F., Rives, P., Gallice, J., Brame, J.: High-speed vehicle guidance based on vision. Control Eng. Pract. **2**(2), 289–297 (1994)
10. Klančar, G., Škrjanc, I.: Tracking-error model-based predictive control for mobile robots in real time. Robot. Auton. Syst. **55**(6), 460–469 (2007)

11. Micaelli, A., Samson, C.: Trajectory tracking for two-steering-wheels mobile robots. IFAC Symp. Robot Control **27**(14), 249–256 (1994)
12. Pivtoraiko, M., Kelly, A.: Efficient constrained path planning via search in state lattices (2005)
13. Qin, S., Badgwell, T.A.: A survey of industrial model predictive control technology. Control Eng. Pract. **11**(7), 733–764 (2003)
14. Schwesinger, U., Pradalier, C., Siegwart, R.: A novel approach for steering wheel synchronization with velocity/acceleration limits and mechanical constraints. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2012)

A Planning Framework for Persistent, Multi-UAV Coverage with Global Deconfliction

Tushar Kusnur, Shohin Mukherjee, Dhruv Mauria Saxena, Tomoya Fukami, Takayuki Koyama, Oren Salzman, and Maxim Likhachev

1 Introduction

Traditional robot coverage is the problem of determining a collision-free path for a robot that covers all points of interest in an environment [10]. *Persistent* coverage seeks to continuously maintain a desired *coverage level* over an environment [9, 17, 22]. This work was sponsored by Mitsubishi Heavy Industries, Ltd. The authors would also like to thank Daniel Feshbach for help with creating figures. In our case, coverage levels degrade over time, thereby increasing the urgency with which points must be revisited. For example, consider a floor-cleaning robot—once a part of the floor is cleaned, more dust will eventually collect over it and thereby decrease its coverage level. There has been a rise in the use of robots to perform tasks cast as persistent-coverage problems, such as environmental monitoring,

---

T. Kusnur · S. Mukherjee (✉) · D. M. Saxena · O. Salzman · M. Likhachev
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: shohinm@andrew.cmu.edu

T. Kusnur
e-mail: tkusnur@andrew.cmu.edu

D. M. Saxena
e-mail: dsaxena@andrew.cmu.edu

O. Salzman
e-mail: osalzman@andrew.cmu.edu

M. Likhachev
e-mail: mlikhach@andrew.cmu.edu

T. Fukami · T. Koyama
Mitsubishi Heavy Industries Ltd., Tokyo, Japan
e-mail: tomoya_fukami@mhi.co.jp

T. Koyama
e-mail: takayuki_koyama@mhi.co.jp

© Springer Nature Singapore Pte Ltd. 2021

G. Ishigami and K. Yoshida (eds.), *Field and Service Robotics*, Springer Proceedings in Advanced Robotics 16, https://doi.org/10.1007/978-981-15-9460-1_32

exploration, inspection, post-disaster assessment, monitoring traffic congestion over a city, etc. [18, 22, 24, 27].

In general, coverage path planning is closely related to the intractable *covering-salesman problem*, where an agent must visit neighborhoods of multiple cities while minimizing travel distance [3]. Extending this to multiple robots requires collision avoidance (with both static and dynamic obstacles), which becomes computationally expensive as the number of robots increases. For persistent coverage, additional algorithmic challenges emerge since robots need to continuously revisit parts of the environment to maintain coverage levels. For a single robot tasked with persistent coverage, there are broadly two decisions to be made:

Q1 Where should the robot go next?
Q2 How should it get there?

We refer to Q1 and Q2 as *goal assignment* and *goal planning*, respectively. Additional questions arise when dealing with multiple robots:

Q3 How do we plan for multiple robots?
Q4 How do we avoid inter-robot collisions?

Assuming a centralized planner, we address Q3 by sequential, decoupled planning instead of planning in the joint state space of all robots for tractability. Specifically, we use *prioritized planning* [8].

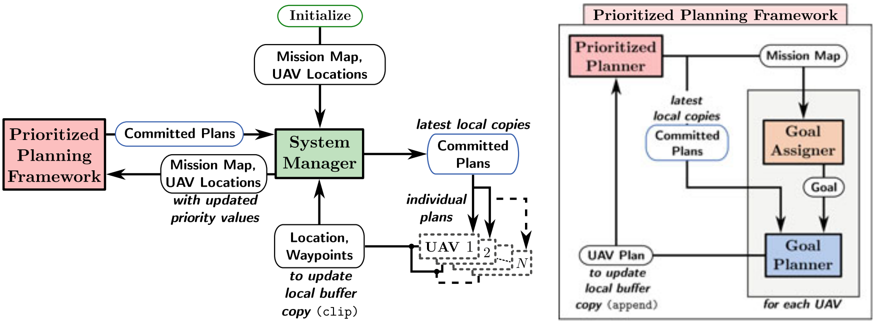
We use the notion of *committed plans* to tie our answers to these questions together. These are defined as parts of plans that robots commit to execute, in contrast to the rest of their plans which may be modified after reassessing the environment. We maintain the invariant of *global deconfliction*—no new committed plan intersects any other existing committed plan in space or time, thereby answering Q4.

In this paper, we present a planning framework to address all the above questions. A block diagram of the complete framework is presented in Fig. 1. To the best of our knowledge, this work is the first to answer all of these questions in unison for real-world, persistent coverage with multiple robots. For our application, these robots are Unmanned Aerial Vehicles (UAVs). The *System Manager* (SM, Sect. 4.1) is our communication interface between the centralized planner and individual UAVs. It is responsible for back-and-forth communication of up-to-date information between the two. The *Prioritized Planner* (PP, Sect. 4.2) is responsible for answering Q3. Thereafter, the *Goal Assigner* (GA, Sect. 4.3) answers Q1 and specifies the location the next UAV should fly to. This is fed to the *Goal Planner* (GP, Sect. 4.4), which plans a kinodynamically feasible path for the UAV to the goal, answering Q2. Part of this plan is appended to the existing committed plan of the UAV. The PP, which operates continuously in a loop, then begins the next planning cycle, answering Q3 again.

We only append a part of computed plans to committed plans of UAVs (further detailed in Sect. 4). This is done so that we maintain committed plans of a pre-specified maximum duration (denoted by $t_{\max}$).[1] There is a trade-off involved in

---

[1] This duration depends on the type and capability of the robots, the number of robots, and the coverage map.


Initialize
Mission Map, UAV Locations
Prioritized Planning Framework
Committed Plans
System Manager
Mission Map, UAV Locations
with updated priority values
latest local copies
Committed Plans
individual plans
Location, Waypoints
to update local buffer copy (clip)
UAV 1
2
N
Prioritized Planning Framework
Prioritized Planner
Mission Map
latest local copies
Committed Plans
Goal Assigner
Goal
Goal Planner
UAV Plan
to update local buffer copy (append)
for each UAV

(a) (b)

Fig. 1 **a** The System Manager (SM) communicates with all UAVs; it sends up-to-date copies of committed plans to corresponding UAVs and updates them using information received from the Prioritized Planner (PP) **b** prioritized planning framework: for each UAV, the Goal Assigner (GA) selects the next goal using the up-to-date map from the PP and the Goal Planner (GP) then plans a feasible path to this goal, which is appended to its committed plan

making $t_{\max}$ too short or too long. Section 4 provides more insight into how and why this arises. We rely on globally deconflicting paths for UAVs since other, more local collision-checking mechanisms have a higher risk of causing *deadlocks*.[2] Some other approaches utilize conflict detection and resolution systems to repair conflicting paths [4]. Our approach solves for deconflicting paths by construction and maintains the invariant of global deconfliction.

In Sect. 2, we contextualize our work among the Orienteering Problem (OP) [30] and the Vehicle Routing Problem (VRP) [29], coverage path planning [10], and frontier-based exploration [31]. Section 3 formalizes the persistent-coverage problem we aim to solve and introduces the notion of priorities over coverage zones. Section 4 provides details about our planning framework and each of its constituent parts. In Sect. 5, we present results of the performance of our framework in both simulated and real-world environments. Finally, Sect. 6 discusses the consequences of some design-related decisions and proposed extensions.

2 Related Work

A majority of existing literature casts the persistent-coverage problem as an instantiation of the OP or the VRP. An OP seeks to determine a path for an agent limited by a time or distance budget that visits a subset of all surveillance sites in an environment and maximizes the sum of associated scores collected. An extension of this to multiple agents is known as the Team Orienteering Problem (TOP) [30]. There are a number of works that apply solutions of the OP to surveillance with aerial vehicles over a small number of surveillance sites [14]. The formulation by Leahy et al. comes closest to our work because of its capability of assigning time windows to each coverage zone as temporal logic constraints [15].

---

[2] Two or more robots are in a *deadlock* if they are not in collision, but executing any valid action for any one of the robots would cause them to collide with another. Hence, they remain stationary.

Given a number of agents at a depot and distances among all surveillance sites and the depot, the VRP seeks to find a minimum-distance tour for each agent such that it visits each site at least once. Michael et al. address persistent surveillance with aerial vehicles as a VRP with modifications to model continuous site visits [11, 25].

The OP and VRP are both variants of the Traveling Salesman Problem (TSP) and thus NP-hard. Moreover, in our case, formulating the problem as a TOP or VRP is not enough since our problem also has an element of 2D coverage. One could consider every discrete part of the environment as a surveillance site in a TOP or VRP to enable 2D coverage, but this is impractical due to increasing computational complexity with the number of surveillance sites.

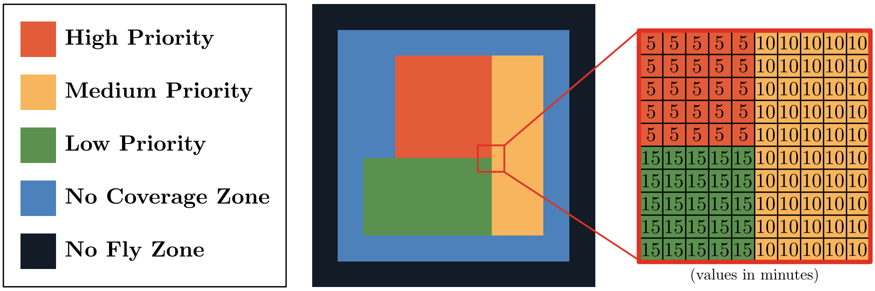
Smith et al. tackle a very similar problem by assuming pre-planned paths, decoupling path planning and speed control for deconfliction [23]. Planning paths for 2D coverage has been extensively studied in robotics [10]. Typical solutions to static coverage involve environmental partitioning via Voroni tessellations and feedback control laws with local collision-avoidance schemes [21]. Environmental partitioning is also a recurring strategy for dynamic and persistent coverage [13]. However, such partitioning restricts individual robots to specific parts of the environment.

For real-world settings, robot path planners must adhere to kinodynamic motion constraints. Thakur et al. solve the TOP and path planning simultaneously to generate 3D (x, y, θ) space-parametrized trajectories, but without temporal deconfliction [28]. Many also formulate planning for persistent surveillance as a mixed-integer linear program (MILP) [5]. However, such MILP formulations introduce limitations that are typically managed by planning for constant-speed paths [1].

A vital part of our framework is to ensure UAVs fly to appropriate goals, so that they simultaneously cover different parts of the environment. We adapt frontier-based exploration to assign such goals to UAVs. Yamauchi et al. first proposed this for single and multiple robots based on an occupancy-grid representation of the environment [31]. Many extensions have been proposed since then, which combine the information gain or expected benefit of the goal and the distance to it, also called next-best-view approaches [2]. Zhu et al. also apply this to exploration and coverage with Micro-aerial Vehicles (MAVs) [32]. We use frontier-based exploration to trade off between cell proximity and cell criticality (defined formally in Sect. 3). To be able to do this online, we use a search-based approach similar to Butzke et al. [6].

3 Problem Formulation

Our problem definition is characterized by a mission map $\mathcal{M}$, and a set of N UAVs $\{U_1, \ldots, U_N\}$, tasked with covering the area represented by $\mathcal{M}$. Each UAV U_k is a kinodynamically constrained system, with an associated coverage or sensor radius r_k.



5	5	5	5	5	10	10	10	10	10
5	5	5	5	5	10	10	10	10	10
5	5	5	5	5	10	10	10	10	10
5	5	5	5	5	10	10	10	10	10
5	5	5	5	5	10	10	10	10	10
15	15	15	15	15	10	10	10	10	10
15	15	15	15	15	10	10	10	10	10
15	15	15	15	15	10	10	10	10	10
15	15	15	15	15	10	10	10	10	10
15	15	15	15	15	10	10	10	10	10

Fig. 2 An example of a mission map $\mathcal{M}$, where each cell is colored according to its lifetime. For example, a green cell with a lifetime of 15 min implies that no more than 15 min should pass between two consecutive times a UAV covers it

Let the cell at row i and column j of $\mathcal{M}$ be $c_{i,j}$. A UAV U_k is at cell $c_{i,j}$ if the projection of its reference point onto the x, y-plane (denoted by U_k^{loc}) lies in cell $c_{i,j}$. A cell is said to be covered by a UAV flying over it if any point on the cell is at most an r_k distance from U_k^{loc}. This effectively assumes a circular sensor footprint of radius r_k centered around U_k^{loc}. Each cell $c_{i,j}$ is associated with two temporal properties—its *lifetime* $\ell(c_{i,j})$ and its *age* $a(c_{i,j})$. The age of a cell is the time passed since the cell was last covered by a UAV, while its lifetime is a desired bound on its age (as shown in Fig. 2).

There are two types of cells in our problem specification: (i) standard cells (that UAVs need to cover) and (ii) obstacle cells (that UAVs cannot fly over). Based on this, we define the following maps:

1. Priority map, $\mathcal{M}^{\mathrm{P}} = \left\{c_{i,j} : c_{i,j} \text{ is a standard cell} \wedge \ell(c_{i,j}) < \infty\right\}$. In other words, $\mathcal{M}^{\mathrm{P}}$ is the set of all standard cells that a UAV will need to cover in finite time since the start of the mission.
2. No-coverage map, $\mathcal{M}^{\mathrm{NC}} = \left\{c_{i,j} : c_{i,j} \text{ is a standard cell} \wedge \ell(c_{i,j}) = \infty\right\}$. In other words, $\mathcal{M}^{\mathrm{NC}}$ is the set of all standard cells in the mission map that a UAV can fly over, but does not need to cover.
3. Obstacle map, $\mathcal{M}^{\mathrm{O}} = \left\{c_{i,j} : c_{i,j} \text{ is an obstacle cell}\right\}$.

The mission map can then be defined as $\mathcal{M} = \mathcal{M}^{\mathrm{P}} \cup \mathcal{M}^{\mathrm{NC}} \cup \mathcal{M}^{\mathrm{O}}$. Figure 2 contains an example of a mission map with three different priority levels. Given a mission map and a set of UAVs, our problem is to compute UAV plans such that the following properties hold:

- P1 Feasibility—the motion of each UAV adheres to its kinodynamic constraints.
- P2 Deconfliction—no two UAVs collide. In our specific setting, all UAVs are constrained to fly at the same altitude, so we say that two UAVs U_p and U_q collide if they come within some predefined distance $d_{\min}$ of each other.
- P3 Persistence—the age of each cell is smaller or equal to its lifetime.
- P4 Flexibility—UAVs can be dynamically added or removed to the system.

P1 and P2 are hard constraints that must always be satisfied. P3 depends on the mission specification—the number of UAVs deployed, their capabilities (speed, sensor radii, etc.), and the mission map (size and lifetime of each cell). Our framework supports P4 without affecting P1 and P2; however, it will affect (help or hinder) our ability to satisfy P3.

4 Approach

Our approach, depicted in Fig. 1, consists of a *System Manager* (SM) which serves as an interface between the centralized planning framework and individual UAVs. It maintains the priority map $\mathcal{M}^{\mathrm{P}}$ by updating locations of individual UAVs.[3] In every planning cycle, the SM passes the three maps to the *Prioritized Planner* (PP). The PP then iterates over all the UAVs in a round-robin fashion. For each UAV U_k, it generates a local copy of $\mathcal{M}^{\mathrm{P}}_k$ that accounts for the committed plans of the other UAVs. This priority map $\mathcal{M}^{\mathrm{P}}_k$ is used by the *Goal Assigner* (GA) to assign a goal (in terms of x, y location) for the UAV U_k. A kinodynamically feasible path to reach this goal is then computed for U_k using the *Goal Planner* (GP).

A key notion used within our framework is that of *committed plans*, which are collision-free plans each UAV is committed to execute. While the GP might compute plans of long durations (if the goal is far away or plan execution requires long, feasible maneuvers), only a portion of this plan will be added to the committed plan, such that it is at most $t_{\max}$ long in time. The value of $t_{\max}$ must be decided based on the mission since the following effects introduce a trade-off:

1. Effects of a high value for $t_{\max}$:
    - The longer the committed plans are, the less reactive UAVs are to any map updates including updates by an operator.
    - The path computed by the GP might take U_k through areas coinciding with parts of the committed plans of $U_{j \neq k}$ before $U_{j \neq k}$ covers them. This results in redundant coverage. The chances of this happening increase as $t_{\max}$ increases.
2. Effects of a low value for $t_{\max}$:
    - The shorter the committed plan, the lesser the chance of deviating from it during execution due to imperfect controllers. However, this increases the chances of UAV U_k executing the committed plan before the next planning cycle ends, leaving the UAV's controller with no plan to execute. We handle this by ensuring that the UAV can execute a stopping maneuver in this situation, but repeated execution of stopping maneuvers is undesirable.
    - Having short committed plans increases the frequency of goal assignment for a UAV. This causes two adverse effects: (i) A UAV may be assigned a new goal

---

[3] Our framework also allows for static obstacles and no-coverage zones to change during a mission, but this is beyond the scope of this paper.

that is different from its previous one, which would cause it to abandon trying to reach the previous goal. (ii) Even if the same goal is assigned, it might lead to "plan thrashing." A UAV's complex dynamics may restrict short plans from letting it make significant progress toward a goal if complicated maneuvers are required.

4.1 System Manager

The System Manager (SM) is the communication interface between the centralized planner and the individual UAVs, shown in Fig. 1 (left): The SM sends the mission map $\mathcal{M}$ to the PP, and it receives the updated committed plans from the PP and sends them out to the respective UAVs. It maintains the priority map $\mathcal{M}^{\mathrm{P}}$ by: (i) resetting the age of any standard cells that have been covered since the last update to zero and (ii) incrementing the age of all other standard cells.

4.2 Prioritized Planner

The Prioritized Planner (PP) is a state machine that controls our centralized planning framework by continuously executing RUNPLANNER from Algorithm 1. During each execution of RUNPLANNER, which we call a *planning cycle*, the PP is responsible for the following tasks:

T1 Update its local copy of the committed plans and $\mathcal{M}^{\mathrm{P}}$ to reflect the most recent information received from the SM.
T2 Determine which UAV will be planned for during this cycle.
T3 Perform a new goal assignment for the UAV and plan a path to it.[4]
T4 Communicate the result of T3 throughout the framework.

T1 is performed in Lines 4–6 in Algorithm 1. The PP clips the part of its local copy of committed plans that have been executed by the UAVs since the last planning cycle, and accounts for the part yet to be executed by resetting the age of all the cells in $\mathcal{M}^{\mathrm{P}}$ that will be covered by the committed plans to zero.

We do not prioritize any one UAV over the others, which corresponds to the PP selecting UAVs in a round-robin fashion, thus achieving T2. While the PP supports a prioritization over UAVs, in our experience, assigning equal priorities in this way showed satisfactory performance.

T3 is achieved through one iteration of the GA-GP loop shown in Fig. 1b. For ease of understanding, we show two successive executions of this loop in Fig. 3. Figure 3b, c corresponds to Lines 9 and 10 for the yellow UAV. Figure 3d, e corresponds to the same lines for the blue UAV.

---

[4] Task T3 is only done if the selected UAV's committed plan is shorter than $t_{\max}$.

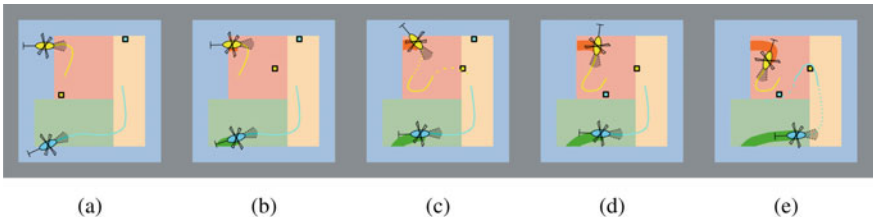

(a) (b) (c) (d) (e)

Fig. 3 Two successive executions of the GA-GP loop (the map $\mathcal{M}$ is colored with 50% opacity). **a** The solid yellow and blue lines show committed plans for both UAVs. **b** A new goal is assigned to the yellow UAV. **c** A path planned is for the yellow UAV (the old committed plan is a dotted line, the new committed plan is a solid line, and the discarded part of the new plan is a dashed line). **d** The same as **b** but for the blue UAV. **e** The same as **b** but for the blue UAV

T4 is performed by Lines 11–13. First, we append to a UAV's committed plan to bring it up to $t_{\max}$ in length. Second, $\mathcal{M}^{\mathrm{P}}$ is updated to account for this new plan (same as in Line 6) by resetting the age of all the cells that will be covered. Finally, the newly committed plan is communicated to the UAV via the SM.

Algorithm 1 Prioritized Planner Loop

```
1:  procedure RunPlanner
2:      $N \leftarrow$ number of UAVs
3:      while true do
4:          Get latest $\mathcal{M} = \{\mathcal{M}^{\mathrm{P}}, \mathcal{M}^{\mathrm{NC}}, \mathcal{M}^{\mathrm{O}}\}$ and $U^{\mathrm{loc}}_{1:N}$ from SM
5:          Update committed plans (local copies) by using $U^{\mathrm{loc}}_{1:N}$          ▷ Clip
6:          Update $\mathcal{M}^{\mathrm{P}}$ with committed plans
7:          for $k \leftarrow 1 : N$ do
8:              if committed plan for $U_k$ is short then
9:                  $G_k \leftarrow$ GetGoal $(\mathcal{M}^{\mathrm{P}})$          ▷ Goal assignment (GA)
10:                 $\pi_k \leftarrow$ GetPlan $(\mathcal{M}, G_k)$          ▷ Goal planning (GP)
11:                 Update committed plan for $U_k$ with $\pi_k$          ▷ Append a portion of $\pi_k$
12:                 Update $\mathcal{M}^{\mathrm{P}}$ with committed plan for $U_k$
13:                 Send committed plan for $U_k$ to SM
```

4.3 Goal Assigner

Recall that the Goal Assigner (GA) selects a goal for each UAV to fly to. The grid $\mathcal{M}^{\mathrm{P}}_k$ is the local copy of the Priority Map ($\mathcal{M}^{\mathrm{P}}$) that belongs to UAV U_k. Each of the cells in the eight-connected grid that represents $\mathcal{M}^{\mathrm{P}}_k$ is a potential goal for U_k. The GA computes a goal location G_{k} for each UAV, accounting for how urgent every cell in $\mathcal{M}^{\mathrm{P}}_k$ is and the location of the UAV U_k. To simultaneously reason about the urgency of the goal and its distance from U_k, we build a graph by connecting a *pseudo-goal*

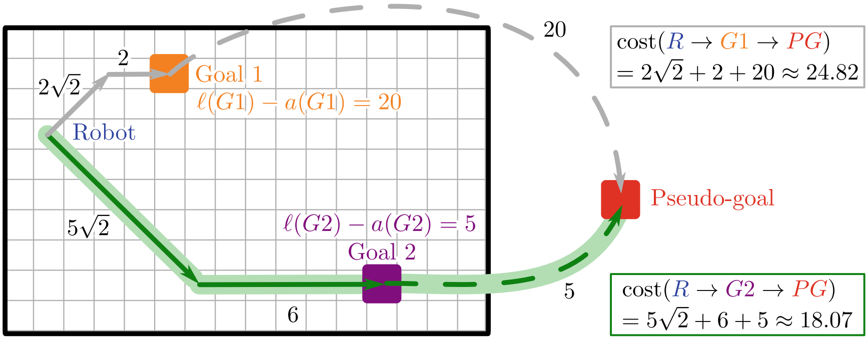


Fig. 4 Multi-goal Dijkstra's search for goal assignment finds the least-cost path to a pseudo-goal connected to all potential goal locations for a robot. It considers the cost of reaching a potential goal from the robot's location, and the priority of a goal location reflected in the cost of the edge connecting it to the pseudo-goal. In the example above, goal G2 is assigned to the robot

to every cell of the grid $\mathcal{M}_k^{\mathrm{P}}$. The pseudo-goal is an "imaginary" state connected by "imaginary" *pseudo-edges* to all cells in the grid as shown in Fig. 4. The cost of these pseudo-edges is proportional to $(\ell(i, j) - a(i, j))$.[5] The costs of the rest of the edges between adjacent cells in the grid are equal to the Euclidean distance between them.

To find an optimal goal G_k, we first find an optimal path from U_k^{loc} to the pseudo-goal on this graph using Dijkstra's search [7]. The optimal path minimizes the sum of the costs of the real edges (which reflects the cost to reach the goal) and the cost of the pseudo-edge (which reflects the priority of the cell that the pseudo-edge connects). The final goal G_k for UAV U_k is the parent of the pseudo-goal on this optimal path.

4.4 Goal Planner

Once a goal is assigned to U_k, the GP computes a kinodynamically feasible path to it via search-based planning on a state lattice [16, 19]. The *state space* for each UAV includes its two-dimensional pose (x, y, θ), linear velocity v, and time t. Assuming double-integrator dynamics for each UAV, we generate an action space consisting of feasible motion primitives. These primitives use cells of size 1 m × 1 m. This discretization is independent of the mission-map discretization. We implicitly construct a graph using the action space during a weighted-A* search to the goal [12, 20]. The search prunes away all transitions that correspond to trajectories that either intersect no-fly zones or collide with the committed plans of other vehicles in space or time. We terminate the search as soon as a state is expanded whose incoming edge (from the predecessor on the found path) covers the goal cell (specifically, the trajectory corresponding to this edge contains a point whose distance to the goal cell is less than r_k). We assign the time taken to execute an action as its edge cost in the graph. While our aim is to plan for minimum time paths, it is also desirable for the UAVs to fly at high speeds whenever possible and avoid stopping. For this reason, we incentivize actions with increasing velocities, penalize actions with decreasing velocities, and heavily penalize hovering.

---

[5] We lower limit these costs by a constant to ensure they are strictly positive. Moreover, since the UAVs' sensors cover multiple cells, the cost of an edge between cell $c_{i,j}$ and the pseudo-goal is the average of $\ell(u, v) - a(u, v)$ for all cells $c_{u,v}$ covered when $U_k^{\mathrm{loc}} = c_{i,j}$.

5 Experimental Setup and Results

We evaluate our framework in both simulations, where we assume perfect state estimation and control, and on real UAVs, which can deviate from their planned paths. The UAVs can withstand winds of speeds up to 30 m/s. Thus, the effect of wind is negligible under normal conditions. Accounting for large deviations from planned paths requires replanning and is part of future work. Our framework uses an identical set of parameters in both cases. We generate motion primitives with a maximum speed of 6 m/s and a maximum turning rate of 6°/s. We avoid adding angular velocity to the state space by ensuring that these primitives start and end at zero angular velocity. Two UAVs are deemed to have *collided* if at any point in time the distance between these 2D locations is less than $d_{\text{min}} = 10$ m. The parameter $r_{\text{k}} = 15$ m defines the circular area directly underneath a UAV that is deemed *covered* for any 2D location of the UAV. We impose a planning timeout of 4 s on the goal planner, which is how long it is given to compute a plan. We use an Euclidean-distance heuristic in the Weighted-A* search [20] with an inflation of 5. Figure 7d and fig:uavmissione shows the two coverage maps used for the experiments presented in this paper.

5.1 Evaluation Metrics

We look at timing statistics for goal assignment (t_{GA}) and goal planning (t_{GP}), number of state expansions, and number of expansions per second.[6] Since we desire continued movement from each UAV, we also look at the amount of time a UAV is stationary on average across the simulation run (t_{stopped}). Additionally, we evaluate our framework's performance with respect to the *criticality* of a cell, defined for any cell (i, j) at timestamp t as $C_t(i, j) = \frac{a_t(i,j)}{\ell(i,j)}$, where $C_t : \mathbb{E}(2) \to \mathbb{R}$ is a time-varying measure of criticality. $C_t(i, j)$ starts with a value of zero and as the cell *ages* over time, $C_t(i, j)$ starts to increase. Once the age of a cell reaches its lifetime—namely, $C_t(i, j)$ equals one—we say that the cell has *expired*. If we average this value across all cells in the map to be covered, we obtain an estimate idea of how well a team of UAVs is covering the areas they have been assigned:

---

[6] These expansions refer to graph-node expansions in the Weighted-A* search.

$$\bar{C}_t = \frac{1}{|\mathcal{M}^{\mathrm{P}}|} \sum_{(i,j)\in\mathcal{M}^{\mathrm{P}}} C_t(i, j) \tag{1}$$

Given this measure of criticality, ideal behavior would be to keep the value of $\bar{C}_t$ constant over the course of a mission, a lower value being better.

5.2 *Simulation Experiments*

Hardware Platform Simulation experiments were performed on a desktop computer running Ubuntu 16.04 and equipped with an Intel Core i7-4790K processor and 20 GB of RAM.

Planner Evaluation We present simulation results on both maps from Fig. 7 for 10 min in Table 1. We observe that by increasing the number of UAVs, goal assignment times are not affected, whereas goal planning times increase significantly since GP must now compute deconflicting plans for a larger number of UAVs. Despite this, the total amount of stationary time spent by a UAV was at most 19.90 s in a 10-min period. Figure 5 shows a plot of $\bar{C}_t$ (Eq. 1) over time during a 30-min mission. In all experiments, our framework is able to keep $\bar{C}_t$ below a value of one except where the number of UAVs was too small given the size of the map (experiment with one UAV and map from Fig. 7). This, along with the fact that all plots plateau over time, implies that cell expiration is regulated on average.

Global Deconfliction Satisfactory global deconfliction relies on each UAV always having access to committed plans that are long enough. As discussed in Sect. 4, size of committed plans can significantly affect performance. We empirically evaluate

Table 1 Results of simulation experiments (values rounded down to the closest integer) and t_{stopped} expressed as a percentage of total mission time

Map	Number of UAVs	Timing (ms)			Path planning		
		t_{GA}	t_{GP}	t_{Total}	Number of expansions	Expansions per second	t_{stopped} (%)
Figure 7d	1	90	223	323	413	215	0
	2	90	796	898	187	226	2
	3	91	1326	1430	286	209	2
Figure 7e	1	92	172	275	28	204	0
	2	91	1163	1265	200	188	<1
	3	91	1433	1536	248	185	3

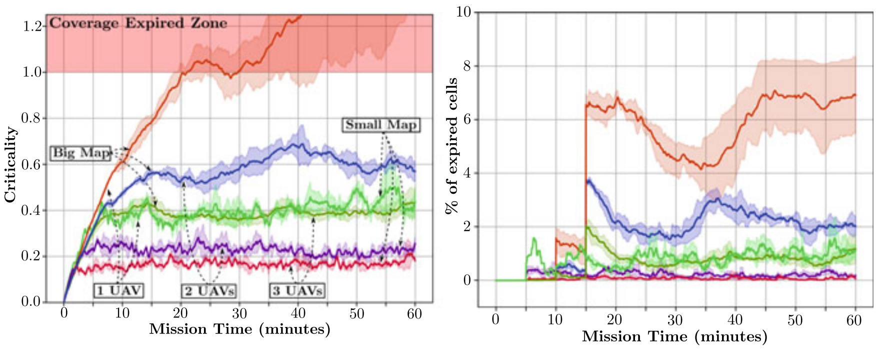
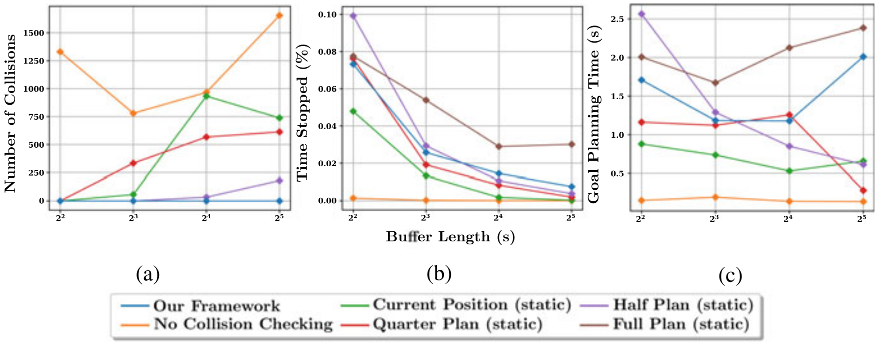


Fig. 5 Left: average criticality of cells in $\mathcal{M}^{\mathrm{P}}$ during simulated experiments. Right: spikes occur at 5, 10, 15 min since these are the three different cell lifetimes in the initial $\mathcal{M}^{\mathrm{P}}$. Colors are consistent across the two plots



(a) (b) (c)

Fig. 6 Effect of changing the committed plan length for different collision-checking mechanisms. Since each curve represents a single mission execution, we do not show confidence intervals. The lines for *Full Plan (static)* and *Our Framework* coincide in **a**

the need and effect of global deconfliction by varying the lengths of committed plans for different collision-checking schemes. These schemes are as follows:

1. Proposed framework: our approach with global deconfliction.
2. No collision checking: The GP performs no collision-checking between UAVs.
3. Current position (static): The GP considers the other UAVs' current positions as static obstacles.
4. Quarter, half, and full plan (static): The GP assumes the first quarter, half, and whole of the committed plans of other UAVs to be static obstacles, respectively.

We plot the number of collisions between UAVs in Fig. 6a, the average time a UAV is stationary in Fig. 6b, and the time taken by the goal planner in Fig. 6c, all against varying lengths of committed plans. Figure 6 shows that our deconfliction scheme results in competitive planning times and also guarantees collision-free UAV move-

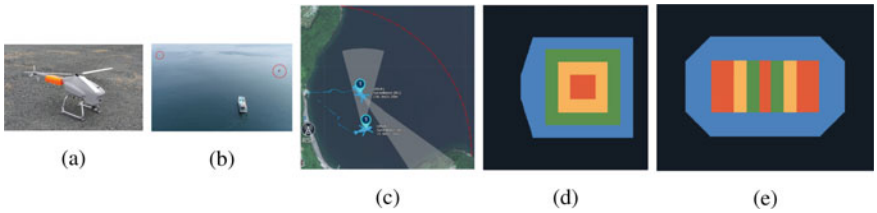


Fig. 7 **a** UAV used in the experiments. **b** The two UAVs (circled in red) executing a mission. **c** GUI showing a satellite image of the mission site along with UAV data overlay. **d** Map used for experiments (big, coverage zone roughly 400 m × 400 m). **e** Map used for experiments (small, coverage zone roughly 200 m × 100 m)

Table 2 Results of real-world experiments averaged over eight runs (rounded down to the nearest integer) and t_{stopped} expressed as a percentage of total mission time (maximum over all runs)

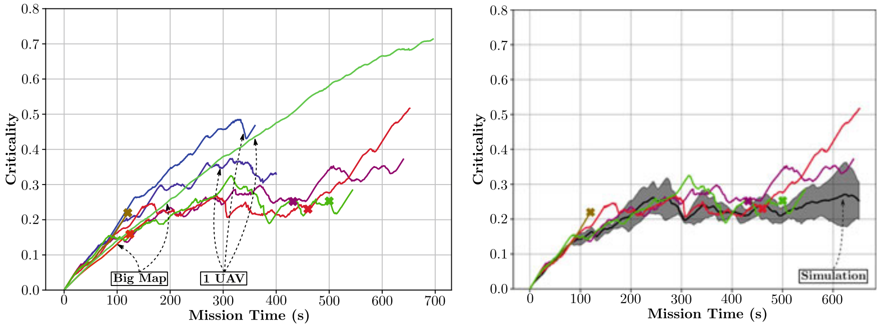
Map	Number of UAVs	Timing (ms)			Path planning		
		t_{GA}	t_{GP}	t_{total}	Number of expansions	Expansions per second	t_{stopped} (%)
Figure 7e	1	105	240	363	37	186	0.00
	2	117	1181	1325	162	152	7.12

ment with negligible stoppage. Other collision-checking schemes are either overly conservative and compute convoluted, long-winding paths, or simply fail to avoid collisions.

5.3 *Real-World Experiments*

Hardware Setup For the real-world experiments, our planning framework was run on a Lenovo T470s laptop running Ubuntu 16.04 and equipped with an Intel Core i7-7600U processor and 20 GB of RAM.

Planner Evaluation We present results from real-world experiments in Table 2. Figure 8a plots $\bar{C}_t$ for eight real-world runs. Additionally, dynamic removal of UAVs from the mission was tested in five of these runs, and the remaining UAVs were left undisturbed. Beyond this point, we are only covering the area with one UAV. Hence, it is expected to see criticality increase. By iteratively planning and executing computed paths, we isolate our planner from the stochasticity of controller-based execution in the real world. Our planner quantitatively performs just as well in the real world as it does in simulation in spite of this stochasticity. This is because our framework seeks latest information about UAV and map statuses from the real world and constantly uses it to repeatedly solve our myopic version of the full problem (as explained in Sect. 4).



(a) (b)

Fig. 8 **a** Coverage criticality over time in the real world: unless explicitly pointed to by arrows, the curves represent an experiment with two UAVs run on the smaller map. A cross (×) on a curve indicates the point in time when one of the two UAVs was removed from the mission. **b** Comparison of all simulation and real-world missions executed in the small map from Fig. 7e by two UAVs. The black curve with confidence intervals corresponds to the simulated experiments

6 Conclusion and Future Work

We present and evaluate a planning framework for real-world, persistent coverage with multiple UAVs. Our framework continuously decides where UAVs should fly and computes kinodynamically feasible, globally deconflicting plans. We evaluate our framework in both simulated and real-world settings. We also motivate and compare global deconfliction with weaker, more local collision-avoidance schemes.

In many practical settings like ours, state spaces are high dimensional and time for deliberation is limited. Planning times can be a bottleneck in these cases and cause delays. While our stopping maneuvers handle such situations, a natural extension is to incorporate anytime planning [26].

In the current framework, the goal assignment is based on priorities and is decoupled from goal planning. This is greedy and not optimal. Better strategies to repeatedly cover previously observed coverage zones can be learned from data and added as macro-actions in the planner.

References

1. Ademoye, T.A., Davari, A.: Trajectory planning for multiple autonomous systems using mixed integer linear programming. In: Proceedings of the Thirty-Eighth Southeastern Symposium on System Theory, pp. 175–179. IEEE (2006)
2. Adler, B., Xiao, J., Zhang, J.: Autonomous exploration of urban environments using unmanned aerial vehicles. J. Field Robot. **31**(6), 912–939 (2014)
3. Arkin, E.M., Hassin, R.: Approximation algorithms for the geometric covering salesman problem. Discrete Appl. Math. **55**(3), 197–218 (1994)

4. Barnier, N., Allignol, C.: Trajectory deconfliction with constraint programming. Knowl. Eng. Rev. **27**(3), 291–307 (2012)
5. Bellingham, J.S.: Coordination and control of UAV fleets using mixed-integer linear programming. Ph.D. Thesis, Massachusetts Institute of Technology (2002)
6. Butzke, J., Likhachev, M.: Planning for multi-robot exploration with multiple objective utility functions. In: Proceedings of IROS, pp. 3254–3259 (2011)
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
8. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. Algorithmica **2**(1–4), 477 (1987)
9. Franco, C., López-Nicolás, G., Sagüés, C., Llorente, S.: Persistent coverage control with variable coverage action in multi-robot environment. In: Proceedings of CDC, pp. 6055–6060 (2013)
10. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
11. Golden, B.L., Raghavan, S., Wasil, E.A.: The vehicle routing problem: latest advances and new challenges, vol. 43. Springer Science & Business Media (2008)
12. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
13. Kapoutsis, A.C., Chatzichristofis, S.A., Kosmatopoulos, E.B.: DARP: divide areas algorithm for optimal multi-robot coverage path planning. J. Intell. Robot. Syst. **86**(3–4), 663–680 (2017)
14. Keller, J.F.: Path planning for persistent surveillance applications using fixed-wing unmanned aerial vehicles. Comput. Oper. Res. (2016)
15. Leahy, K., Zhou, D., Vasile, C.I., Oikonomopoulos, K., Schwager, M., Belta, C.: Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints. Auton. Robots **40**(8), 1363–1378 (2016)
16. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. IJRR **28**(8), 933–945 (2009)
17. Mellone, A., Franzini, G., Pollini, L., Innocenti, M.: Persistent coverage control for teams of heterogeneous agents. In: Proceedings of CDC, pp. 2114–2119 (2018)
18. Nedjati, A., Izbirak, G., Vizvari, B., Arkat, J.: Complete coverage path planning for a multi-UAV response system in post-earthquake assessment. Robotics **5**(4), 26 (2016)
19. Pivtoraiko, M., Kelly, A.: Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In: Proceedings of IROS, pp. 3231–3237 (2005)
20. Pohl, I.: Heuristic search viewed as path finding in a graph. Artif. Intell. **1**(3–4), 193–204 (1970)
21. Schwager, M., Rus, D., Slotine, J.J.: Decentralized, adaptive coverage control for networked robots. IJRR **28**(3), 357–375 (2009)
22. Smith, R.N., Schwager, M., Smith, S.L., Jones, B.H., Rus, D., Sukhatme, G.S.: Persistent ocean monitoring with underwater gliders: adapting sampling resolution. JFR **28**(5), 714–741 (2011)
23. Smith, S.L., Schwager, M., Rus, D.: Persistent robotic tasks: monitoring and sweeping in changing environments (2011). arXiv:1102.0603
24. Srinivasan, S., Latchman, H., Shea, J., Wong, T., McNair, J.: Airborne traffic surveillance systems: video surveillance of highway traffic. In: Proceedings of the ACM 2nd International Workshop on Video Surveillance & Sensor Networks, pp. 131–135 (2004)
25. Stump, E., Michael, N.: Multi-robot persistent surveillance planning as a vehicle routing problem. In: IEEE International Conference on Automation Science and Engineering, pp. 569–575 (2011)
26. Sun, X., Koenig, S., Yeoh, W.: Generalized adaptive A*. In: Proceedings of AAMAS, pp. 469–476. International Foundation for Autonomous Agents and Multiagent Systems (2008)
27. Teixeira, L., Alzugaray, I., Chli, M.: Autonomous aerial inspection using visual-inertial robust localization and mapping. In: Proceedings of FSR, pp. 191–204. Springer (2018)
28. Thakur, D., Likhachev, M., Keller, J., Kumar, V., Dobrokhodov, V., Jones, K., Wurz, J., Kaminer, I.: Planning for opportunistic surveillance with multiple robots. In: Proceedings of IROS, pp. 5750–5757 (2013)
29. Toth, P., Vigo, D.: The vehicle routing problem. SIAM (2002)

30. Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: a survey. Eur. J. Oper. Res. **209**(1), 1–10 (2011)
31. Yamauchi, B., et al.: Frontier-based exploration using multiple robots. Agents **98**, 47–53 (1998)
32. Zhu, C., Ding, R., Lin, M., Wu, Y.: A 3D frontier-based exploration tool for MAVs. In: Proceedings of ICTAI, pp. 348–352 (2015)